Table of Contents

1 End of Day Info Processes/Scripts and Activities

Overview

The End of Day Window in the System Administration module permits the user to initiate the end of day processing procedures required at different host computer sites. Since the AIM system architecture is a distributed client/server system that will utilize 21 Local Agency/Clinic servers and one Central Operations server, this processing runs on a nightly basis (every night of the week, excluding Sunday) to:

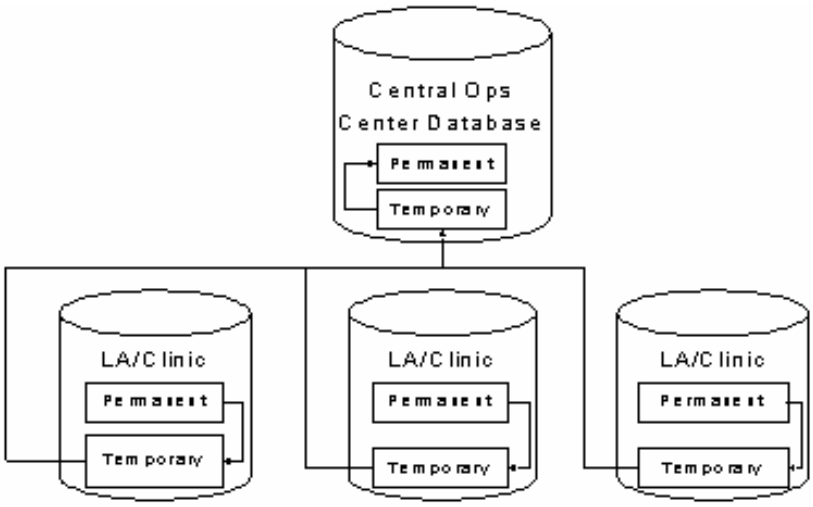
- synchronize data at the Local Agency/Clinic Servers to the data at the Central Operations Server
- perform automated participant data updates such as category changes and appointment status
- delete unnecessary information to improve system performance
- generate log files
- send/receive food instrument issuance and reconciliation information with the banking intermediary FSMC
- archive and retrieve archived participant records
- update caseload information at the local and central database levels
- update vendor peer group averages data

This screen allows the user to initiate the process manually, though each Local Agency/Clinic's AIM database can have a pre-programmed time that the End of Day process initiates automatically.

Flow of Data through End of Day

The End of Day processing is shown in a table format at the end of this narrative. It provides the reader with a logical representation of the processing and scripts involved in the End of Day Process, separated as much as possible by the location at which they occur (Local Agency/Clinic, Central or Bank sites). In the physical running of the End of Day processing, the scripts don't run in this order. Some scripts run concurrently in the effort to accommodate time constraints. To view a detail of the order of the scripts, please see Appendix D - End of Day Script Run Order.

As discussed above, the End of Day processing runs in order to perform several automated updates, including synchronizing the data on the AIM distributed WIC system, so that the Local Agency/Clinic servers, Central server and Banking Intermediary (FSMC) all have updated information.



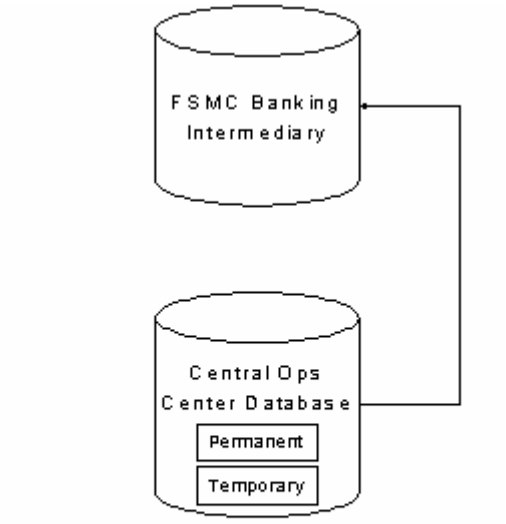
Step 1 - Data is transferred from permanent tables to temporary tables at the Local Agency/Clinic level. From there files are zipped and sent up to a temporary table at the Central Server level. Here the files are moved into a permanent table at the Central Server.

The processing begins at the Local Agency/Clinic Servers with the system performing updates to the Local Agency/Clinic data, such as: performing automatic category changes and marking participant appointments as Missed or Kept. These updates are done within the LA/Clinic servers to the permanent system tables that reside on each server database. When the updates are done, the system gathers participant and food instrument information for transmission to the Central Server. The information is gathered together by copying the updated permanent food instrument and participant table information in the LA/Clinic servers, then copying the tables into temporary output tables. The tables are listed in a parameter file and are exported to a dmp file with that particular Local Agency's number as the filename and copied to the central server to be imported into the Central Operations Server database.

Coinciding with the Local Agency/Clinic servers performing updates, the Central server begins to delete unnecessary information on the Central server through the deletion of records from permanent tables stored on the Central Operations Center database. The system then begins to prepare archive retrieval data at the Central database by retrieving the archived vendor and participant records from tape, storing them in permanent tables on the Central Server, and then copying them into temporary tables in preparation for transfer to the LA/Clinic Servers. Also, the system compiles food instrument data (for Compliance Buys) and vendor information for transmission to the Banking Intermediary. The system compiles the food instrument data and the vendor information into separate temporary text files in preparation for transmission. New temporary files are created each time End of Day runs.

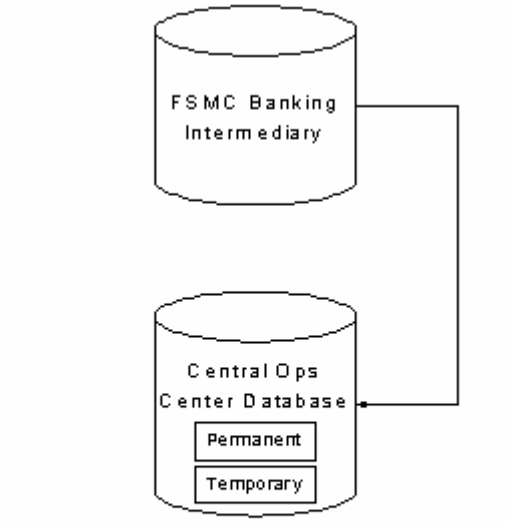
Both the updates at the LA/Clinic Databases and the compilation of information at the Central Operations Server Database haven't caused any information to move across the Arizona WIC WAN. At this point, the information is simply being formatted and prepared for transmission.

At a predetermined time (i.e. 7:30pm the Central Server searches for the dmp file by looking for the particular Local Agency number prefix and then verifying its existence by returning a value of 'True' to the search procedure. The Central server then performs consolidation and update processing of caseload, financial, food instrument and participant data by comparing the updated information to the Central Operations Server database permanent files and copying the more recent data from the temporary tables to the permanent tables .



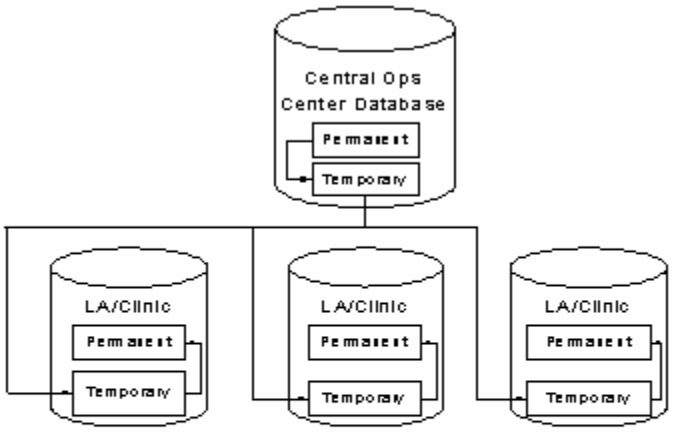
Step 2 - Files are sent from the Central Server up to the FSMC

When the updating and processing of data at the Central Server is completed in the previous step the Central Operations Server Database compiles the new issuance and void data received from the LA/Clinics into an output temporary file. This food instrument information, along with the vendor information gathered earlier just after End of Day initiation is ready to be transmitted to the bank (this time without the benefit of a zip format). The system polls the banking intermediary and sends the updated information.



Step 3 - Files are sent back down from the FSMC to the Central Server

While performing updates at FSMC, the system automatically picks up information at the bank (see Appendix B) and then consolidates and updates payment record and vendor information at the Central server. The updated temporary files received from FSMC are also in an unzipped format and the system compares the updated temporary information against the permanent tables and performs updates accordingly.



Step 4 - Data is transferred from a permanent table to a temporary table at the Central Server. From there files are zipped and sent back down to temporary tables at the Local Agency/Clinic level. Here the files are moved back into permanent tables.

After finalizing consolidation of the payment and vendor information at the Central site, the system prepares a dmp file containing updated caseload and food instrument redemption and rejection information. As well, archived client information that was requested is included in the dmp file. The dmp file is appropriately named based upon the LA/Clinic numbers. When the LA/Clinic receives its dmp file, the LA/Clinic database, compares the temporary tables within it to the permanent tables on the LA/Clinic database and performs updates accordingly.

1. The Oracle export (hot backup) is done at 2:30am from Tuesday until Saturday. It backs up the agency database, and shuts down the database for cold backup.
2. The NT Backup runs at 3:00am from Tuesday until Saturday. It backs up the c:\ and d:\ drives of the Local Agency server so that all the data is available on the backup tape. Tapes are then stored on and offsite according to Arizona policy and procedures.
3. The “at” job for rebuilding indexes is done on every Sunday at 7:00 am.
4. The “at” job for analyzing AIM schema is run on every Monday.
5. Database startup job runs at 5:30am from Tuesday until Saturday.

The End of Day processing will perform the following tasks utilizing the scripts listed each time it runs:

Processes	Script	LA	Central	Bank
Perform backup	aimXX_backup.cmd shutaimXX.cmd	x		
Automatic termination of participants no longer eligible (omit if done during clinic day)	EA1_TERM_EOD.SQL, EA1_TERM_WL_CLIENTS.SQL	x		
Perform automatic category changes	EA1_CAT_EOD.SQL	x		
Mark appointments as Missed or Kept	EA1_APPT_EOD.SQL EA1_NEW_APPT_EOD.SQL EA1_NEW_CLASS_EOD.SQL	x		
Delete information no longer used by the system (Includes participants with no Certification Record after 60 days on system)	EA1_OTHER_PURG.SQL,	x		
Set exclusively breastfeeding participants to active status	EA1_IEN_ACTIVE.SQL	x		
Void FIs not deposited after the stale date	EC1_STALE_FI.SQL		x	
Gather updated participant information, new issuance information, void information, requests for transfer, and outreach org. information	EA1_TAB_UP.SQL EA2_EXPORT_TAB.TXT	x		
Delete information no longer used by the system	EC1_OTHER_PURG.SQL	x	x	
Compile direct payment and revalidated FI information and send to bank	EC1_CTRL_CHCK.SQL		x	x
Compile vendor information to be sent to the bank	EC1_VENDOR_FSMC.BAT, EC1_PGA_FSMC.SQL, EC1_VENDOR_FSMC.SQL,		x	
Consolidate participant information	EC2_SP_TRNC.SQL, EC2_TRNC.SQL, EC2_DELTRIG.SQL, EC2_DELTRIG_AU.SQL, EC2_IN_TAB.SQL, EC2_TAB_UP.SQL EC2_DEL_CL_FUT_RECS.SQL		x	
Financial - process totals and store in F_CASELOAD table. Totals are based on priority, language, caseload type, ethnic group, and poverty level	EC2_CL_REC_PROCESS.SQL		x	
Update caseload assignment information from the clinics	EC2_SYNC_CLINIC_ASSIGNMENT.SQL		x	
Consolidate and update all food instrument data to calculate FI obligation value	EC2_FOOD_UP.SQL, EC2_FI_UPD.SQL		x	
Search for potential dual enrollment participants	EC2_DUAL_ENROLL.SQL		x	
Compile new issuance and void information to send to the bank	EC2_ISSUE_FSMC.SQL		x	x
Create consolidated record of all FI issuance and redemption information and create vendor payment	EC4_BANK.SQL, EC4_BNK_POST.SQL		x	

records				
Create date of current food funds available from all sources as well as current food obligations and expenditures to establish current cash position	EC4_CASHFLOW_UPD.SQL		x	
Financial (end of month) - Populate F_INCOME to POVERTY table. These values are used in populating the caseload table	EC4_MON_END.SQL		x	
Vendor (end of month) calculate and update peer group averages	EC4_VEN_PEER_FI.SQL EC4_VEN_LOC_AGCY.SQL		x	
Vendor run and print analysis factor reports	EC4_VEN_PRICE.SQL EC4_VEN_RISK_HISTORY.SQL		x	
Generate dual enrollment report to be sent to agencies	EC4_DUAL_RPT.BAT EC4_DUAL.RPT		x	
Gather new/updated data to be sent to the agencies	EC4_TRNC.SQL EC4_TAB_DOWN.SQL		x	
Prepare archival retrieval data	EC1_PRG_ARCHV.SQL, EC4_RETRIEVE.SQL, EC4_INS_RETR_CLI.SQL EC4_INS_RETR_VEN.SQL		x	
Print out status logs	EC_PRINT.BAT		x	
Consolidate information from central	EA3_TRNC.SQL, EA3_PRE.SQL, EA3_DELTRIG.SQL, EA3_IN_TAB.SQL, EA3_TAB_UP.SQL		x	
*Clinic serial number replenishing	EC4_CL_SER_REP.SQL		x	
Update caseload assignment information from the clinics	EA3_SYNC_LA_ASSIGNMENT.SQL	x		
Send redemption/rejection information to agencies	EA3_PROC_FI.SQL	x		
Transmit archival parts. and vendors and update records at agencies	EA3_PRG_ARCHV.SQL, EA3_RETRIEVE.SQL EA3_INS_RETR_CLI.SQL EA3_INS_RETR_VEN.SQL	x	x	
*Vendor monthly reports			x	
*Vendor quarterly reports			x	
Print out status logs of the process	EA_PRINT.BAT	x		

*New End of Day processes

Oracle notation for schema and table names:

A table name preceded by "AIM." indicates that it is in the schema that holds permanent tables. The schema name is the same for all agencies and the central server.

A table name preceded by "EODADM." indicates that it is in the schema that holds temporary tables, starting with A_, E_, and R_, as well as the DEL_STATEMENTS and EOD_CONTROLS tables.

- The tables whose names begin with **A_** (not the appointment scheduler tables) are those that are populated at the Central Server, to be sent to the Agency Servers.
- The tables whose names begin with **E_** are those that are populated at the Agency Servers to be sent to the Central Server.
- The tables whose names begin with **R_** are used for Participant data for newly-archived participants.
- The **DEL_STATEMENTS** table contains the actual SQL delete statements that are issued at the Central Database, which must be executed again at the Agencies, and vice versa.
- The **EOD_CONTROLS** table contain information about the dates that the EOD runs have taken place: to_date (the date/time that the EOD process started), from_date (the date/time that the EOD process ended), good_proc_date (the last date on which a successful EOD run was performed).

A processing description with the **Count (*)** command in it will provide a count of the number of records meeting a select statement's WHERE clause.

In the Selection and Processing sections in the EOD portion of the DTSD, occasionally a table "**alias**" is used. This is a short name that is defined so that it can be used instead of the long table name. In the following example, the alias for the I_BANK_REPORTS table is IBR:

```
select    ibr.serial_number,
          ifi.ifit_fi_type_code
from      i_bank_reports ibr,
          i_food_intstruments ifi
where     ibr.serial_number = ifi.serial_number
```

1.1 Perform back-up

Overview

The system performs hot and cold backups of the application and database from Tuesday until Saturday at 2:30am and 3:00am respectively.

Processes:

1.1.1 ShutaimXX.bat

The system initiates the following NT commands on each Local Agency server to perform the hot backup and shutdown of the database.

Exp system/manager parfile=d:\oracle\backup\expaim01.par
C:\arizona\817\bin\sqlplus internal/oracle @d:\oracle\backup\aim_shut.sql

1.1.2 AimXXX_backup.bat

The system initiates the following NT commands on each Local Agency server to perform a cold backup. The Cold backup includes incremental as well as a full backup.

C:\winnt\system32\ntbackup backup c:\arizona d:\oracle c:\orant d:\arizona /b /d "Complete backup of all oracle directories" \e \hc:on /l "d:\aim\eod\agency\logs\full_bkp.log" /t normal

C:\winnt\sysem32\ntbackup backup c:\ d:\ /a /d "Incremental backup of all drives" /hc:on /t incremental /l "d:\aim\eod\agency\logs\inc_bkp.log"

1.2 EA1.BAT

Overview

This batch file begins the End of Day processing at the Agency level.

Processing

On every agency server this script is run as part of the end of day process.

This script is also called from the Systems Administration End of Day form when the proceed button is pressed if the AGCY_SEQ is not equal to ‘00'.

The system sets the environmental variables for the particular Local Agency reading them from the EA1.INI file and places them in a temporary file.

The following SQL scripts are then executed:

EA1.SQL which updates the EOD_CONTROLS.TO_DATE to sysdate and sets the environmental variables. It then runs the following EA1 scripts:

- EA1_TERM_EOD.SQL
- EA1_CAT_EOD.SQL
- EA1_CAT_SYNC_EOD.SQL
- EA1_APPT_EOD.SQL
- EA1_NEW_CLASS_EOD.SQL
- EA1_NEW_APPT_EOD.SQL
- EA1_PURGE_OLD_APPT.SQL
- EA1_PURGE_NEW_APPT.SQL
- EA1_OTHER_PURG.SQL
- EA1_TERM_WL_CLIENTS.SQL
- EA1_FOOD_CHNG.SQL
- EA1_IEN_ACTIVE.SQL
- EA1_TRANSFER.SQL
- EA1_STATE_CLIENTS_UPD.SQL
- EA1_CSF_OU_FIX.SQL

EA1_TAB_UP.SQL which extracts client information that has been changed or created and exports these tables.

The tables are then dumped into a .dmp file with format %. dmp where the wildcard represents the Local Agency ID, and a completion flag is created used to indicate the completion of the copying process with format %.don where the wildcard represents the Agency ID.

The script then transfers the dump files to central ftp_server using FTP protocol.

The script finishes by updating the EOD controls by first searching for the completion flag and then running EA2.SQL which updates the EOD_CONTROLS FROM_DATE and GOOD_PROC_DATE to the sysdate.

1.2.1 EA1_SOFTWARE_UPD.BAT

This script updates new software at agencies from the Central Server.

1.2.2 EA1.SQL

Overview

Executes SQL scripts for EA1.BAT.

1.2.2.1 Inputs

eod_controls
env_variables

1.2.2.2 Selection

env_variables
where code = 'EOD_BY'

1.2.2.3 Processing

update eod_controls, setting TO_DATE to the system date
update env_variables, setting env_value to 'EA1' where code = 'EOD_BY'

run SQL script EA1_TERM_EOD
run SQL script EA1_CAT_EOD
run SQL script EA1_CAT_SYNC_EOD
run SQL script EA1_TRANSFER
run SQL script EA1_APPT_EOD
run SQL script EA1_NEW_CLASS_EOD
run SQL script EA1_NEW_APPT_EOD.SQL
run SQL script EA1_PURGE_OLD_APPT.SQL
run SQL script EA1_PURGE_NEW_APPT.SQL
run SQL script EA1_OTHER_PURG
run SQL script EA1_TERM_WL_CLIENTS
run SQL script EA1_FOOD_CHNG
run SQL script EA1_IEN_ACTIVE
run SQL script EA1_TRANSFER.SQL
run SQL script EA1_STATE_CLIENTS_UPD.SQL
run SQL script EA1_CSF_OU_FIX.SQL

1.2.2.4 Outputs

eod_controls
env_variables

Log file: EA1_SQL.LOG

1.3 Automatic termination of clients no longer eligible

1.3.1 EA1_TERM_EOD.SQL

During the End of Day process automatic terminations* are performed as follows:

- *• The WIC participant is postpartum and today's date is 6 months after the actual delivery date
- *• The CSF participant is postpartum and today's date is the last day of the calendar month of the Actual Delivery date + 12 calendar months
- *• The WIC participant is breastfeeding and today's date is 12 calendar months after the Actual Delivery date
- *• The CSF participant is breastfeeding and today's date is the last day of the calendar month of the Actual Delivery date + 12 calendar months
- *• The WIC participant is 5 years of age and today's date is the last day of the calendar month in which the participant turned 5 years of age
- *• The CSF participant is 6 years of age and today's date is the last day of the calendar month in which the participant turned 6 years of age
- *• The WIC or CSF participant's current certification period has expired and there is not a new certification period

Participants that have missed three consecutive appointments are no longer automatically terminated during the End of Day process.

*For all Participant records terminated all future appointments, except for certification and re-certification appointments, for the participant are deleted during the End of Day process.

The following table outlines the term_code's and descriptions:

Term_code	Description
A	NO NUTRITIONAL RISK FOUND
B	NOT INCOME ELIGIBLE
C	BREASTFEEDING WOMAN NO LONGER BREASTFEEDING
D	RISK FACTORS RESOLVED-GRADUATED- SEEN FOR RECERTIFICATION
E	VOLUNTARY WITHDRAWAL
F	DUPLICATE RECORD
G	MOVED OUT OF STATE
H	LOST TO FOLLOW-UP
I	DEATH
J	MOVED OUT OF CURRENT LOCAL AGENCY
K	MISSED 2 FOOD BOX PICK-UPS
L	XX-DO NOT USE
M	ERROR IN TERMINATION DATE
N	NOT PRIORITY NUTRITIONAL RISK-SEEN FOR RECERTIFICATION
O	NOT CURRENTLY SERVING PRIORITY IDENTIFIED/REFUSED WAIT LIST
P	ABUSE OR THREAT TO ANYONE CONNECTED TO WIC/CSF PROGRAM
Q	MISUSE OF FOOD INSTRUMENTS
R	SALE OR EXCHANGE OF FOOD OR FOOD INSTRUMENTS
S	INTENTIONAL MISREPRESENT OR WITHHOLD FACTS TO OBTAIN BENEFIT
T	RECEIPT OF CASH OR CREDIT TO PURCHASE UNAUTHORIZED FOODS
U	NOT SEEN FOR RECERTIFICATION
V	NOT RECERTIFIED. AUTO TERM FOR INFANT, CHILD, OR WOMAN
W	WOMAN AT 6 WEEKS POSTPARTUM
X	PREGNANT WOMAN AT 3 MONTHS PAST EXPECTED DELIVERY DATE
Y	NON-BREASTFEEDING WOMAN AT 6 MONTHS PAST ACTUAL DELIVERY

Z	BREASTFEEDING WOMAN AT ONE YEAR PAST ACTUAL DELIVERY DATE
1	CHILD FIVE YEARS OLD
2	TRANSFERRED TO ANOTHER AZ WIC LOCAL AGENCY
3	MISSED 4 MONTHS OF FOOD PACKAGES
4	CHILD SIX YEARS OLD
5	CURRENTLY PARTICIPATING IN CSFP
6	CURRENTLY PARTICIPATING IN WIC
8	CATEGORICALLY INELIGIBLE
9	CATEGORY CHANGE
0	CERTIFICATION PERIOD ENDED

1.3.1.1 Inputs

eod_controls
aas_appt_clients
aas_appt_items
aas_appointments
aas_class_families
c_cert_term_reasons
c_certifications
c_client_groups
c_client_ne_topics
c_client_services
c_client_svc_ne_materials
c_clients
c_family_economic_units
f_wait_lists

1.3.1.2 Selection

Select c_certification records where termination_date is null and check_out_flag = 'N'

1.3.1.3 Processing

Main Routine

For each C_CERTIFICATIONS record

```
cat_code    := C_CERTIFICATIONS.CAT_CATEGORY_CODE
c_e_date    := C_CERTIFICATIONS.CERT_END_DATE
r_stat      := C_CLIENTS.REC_STATUS
term_code := null;
```

- Determine the Termination Reason

if cat_code = 'P' (postpartum) and C_CERTIFICATIONS.ACTUAL_DELIVERY_DATE is more than six months ago (WIC) or (C_CERTIFICATIONS.ACTUAL_DELIVERY_DATE is twelve months ago(CSF) and today is the last day of the month) then
term_code := 'Y'

if cat_code in ('EN','PN') (nursing woman) and
C_CERTIFICATIONS.ACTUAL_DELIVERY_DATE is more than 12 months ago and today is the last day of the month for CSF participants then
term_code := 'Z'

if term_code is null and cat_code in (''IEN','IFF','IPN','C1','C2','C3','C4') (child)

(The child over 5 years should take place on the last day of the month following the end of the birthdate month.)
wic_max_date := LAST_DAY(ADD_MONTHS(C_CLIENTS.BIRTH_DATE,60)

if (C_CERTIFICATIONS.PROGRAM = 'WIC' and wic_max_date is today or earlier)
term_code := '1'

if term_code is null and cat_code in ('C5')
(The child over 6 years should take place on the the last day of the month following the end of the birthdate month.)

csf.max_date:= Last_day (add_months (C_clients.birth_date,72)
if (C_certifications.program= 'CSF' & csf.max_date is today or earlier)
term_code = '4'

if c_e_date is today or earlier (the certification end date is today or earlier)
term_code := 'V'

if cat_code in ('P', 'EN', 'PN') and programis CSF:
csf.max_date;= last_day (add.months (actual_delivery _date,12))

if csf.max_date is today or earlier
term code = '8'

-- Terminate the certification period, and if the client does not have a future
-- certification period then terminate the client

If term_code is not null

If C_CERTIFICATIONS.CERT_START_DATE > today and TERMINATION_DATE is null
Active_client := 'Y' (The client has a future certification period)

Else
update C_CLIENTS set TR_TERMINATION_CODE = term_code, REC_STATUS = 'I',
PREV_REC_STATUS = r_stat
Active_client := 'N'
End if;

update c_certifications set termination_date = current date, wait_list_flag = 'N'
update f_wait_lists set active_flag = 'N'
insert term_code into c_cert_term_reasons

- Remove all future appointments except Certification and Re-Certification appointments

If active_client = 'N'
For each future AAS_APPOINTMENTS record that is not for a Certification or Re-Certification Service loop
Delete the A_AAS_APPT_ITEMS records for the AAS_APPOINTMENTS record

Delete AAS_APPT_CLIENTS for the AAS_APPOINTMENTS record.

Update the AAS_APPOINTMENTS record set CFEU_FAMILY_ID to null and
AAS_AS_STATUS_CODE = ‘P’
Delete AAS_CLASS_FAMILIES

For each C_CLIENT_SERVICES record where
APPT_FLAG = 'Y' and
SERVICE_DATE > Today's date and
SRV_SERVICE_CODE not in ('CERT','RECERT') loop

Delete the C_CLIENT_SVC_NE_MATERIALS records for each of the
C_CLIENT_NE_TOPICS records for the C_CLIENT_SERVICES record
Delete the C_CLIENT_NE_TOPICS records for the C_CLIENT_SERVICES record
Delete the C_CLIENT_SERVICES record

End loop
End loop

-- Remove the Client from Class appointments,

For each AAS_APPOINTMENTS record where FAMILY_ID = the family id loop

For each AAS-CLASS_FAMILIES record where CFEU_FAMILY_ID = the family id
Delete AAS_CLASS_FAMILIES record
Update AAS_APPOINTMENTS to set appointment status as ‘Pending’

End loop
End loop

Insert into ENV_VARIABLES (code, env_value, date_created, created_by, date_modified,
modified_by, note)
values (C_CLIENTS.CLIENT_ID, term_code, sysdate, USERID, null, null,'termscript');

end if active_client = 'N'

End loop

End

1.3.1.4 Outputs

aas_appointment
aas_appt_items
aas_appt_clients
aas_class_families
c_certifications
c_client_groups

c_client_ne_topics
c_client_services
c_client_svc_ne_materials
c_clients
f_wait_lists
c_cert_term_reasons

1.3.2 EA1_TERM_WL_CLIENTS.SQL

Overview

This script updates client records to indicate that they are terminated if the clients are on the waiting list for more than the allotted time requirements defined in the F_CONTROLS table for the waiting list. (This does not apply to clients classified in check out clinic status.)

1.3.2.1 Inputs

eod_controls
aas_appointment
aas_appt_items
aas_appt_clients
aas_class_families
a_appointments
a_appt_appt_items
a_appt_topic_materials
a_appt_topics
a_clinic_days
a_groups
c_cert_term_reasons
c_certifications
c_client_groups
c_client_ne_topics
c_client_services
c_client_svc_ne_materials
c_clients
c_family_economic_units
f_wait_lists

1.3.2.2 Selection

Select client ID (ccl_cc_client_id) from f_wait_lists where date added to wait list (date_on_wait_list) <= (effective date - F_CONTROLS.WAIT_LIST_PERIOD)

1.3.2.3 Processing

Main Routine

For each F_WAIT_LISTS record

r_stat := C_CLIENTS.REC_STATUS
term_code := ‘V’

-- Terminate the certification period, and if the client does not have a future
-- certification period then terminate the client

If term_code is not null

If C_CERTIFICATIONS.CERT_START_DATE > today and TERMINATION_DATE is null
Active_client := ‘Y’ (The client has a future certification period)

Else
update C_CLIENTS set TR_TERMINATION_CODE = term_code, REC_STATUS = ‘I’,
PREV_REC_STATUS = r_stat
Active_client := ‘N’
End if;

update c_certifications set termination_date = current date, wait_list_flag = ‘N’
update f_wait_lists set active_flag = ‘N’
insert term_code into c_cert_term_reasons

- Remove all future appointments except Certification and Re-Certification appointments

If active_client = 'N'
For each future A_APPOINTMENTS record that is not for a Certification or Re-Certification Service loop
Delete the A_APPT_APPT_ITEMS records for the A_APPOINTMENTS record
Delete the A_APPT_TOPIC_MATERIALS records for each of the A_APPT_TOPICS for the A_APPOINTMENTS record
Delete the A_APPT_TOPICS records for the A_APPOINTMENTS record
update the A_APPOINTMENTS record set CCS_CC_CLIENT_ID to null and ATS_ATTEND_STATUS_CODE = ‘P’ and GRO_GROUP_ID = null and WE_WORK_EVENT_ID = null and CCS_ID = null

For each future AAS_APPOINTMENTS record
Delete corresponding AAS_APPT_CLIENTS
Delete AAS_APPT_ITEMS
Mark AAS_APPOINTMENTS aas Pending if this client is the only member for that family appointment.

For each C_CLIENT_SERVICES record where
APPT_FLAG = 'Y' and
SERVICE_DATE > Today's date and
SRV_SERVICE_CODE not in ('CERT','RECERT') loop

Delete the C_CLIENT_SVC_NE_MATERIALS records for each of the C_CLIENT_NE_TOPICS records for the C_CLIENT_SERVICES record

Delete the C_CLIENT_NE_TOPICS records for the C_CLIENT_SERVICES record
Delete the C_CLIENT_SERVICES record

End loop
End loop

-- Remove the Client from group appointments, and remove the group if there are no more -- clients in the group

For each C_CLIENT_GROUPS record where CC_CLIENT_ID = the client id loop
If there is a future A_APPOINTMENTS record where GRO_GROUP_ID = C_CLIENT_GROUPS.GROUP_ID
Delete the C_CLIENT_GROUPS record

If there are no more clients in the C_CLIENT_GROUPS table for this group

For each future A_APPOINTMENTS record where GRO_GROUP_ID = C_CLIENT_GROUPS.GROUP_ID loop
Delete the A_APPT_APPT_ITEMS records for the A_APPOINTMENTS record
Delete the A_APPT_TOPIC_MATERIALS records for each of the A_APPT_TOPICS for the A_APPOINTMENTS record
Delete the A_APPT_TOPICS records for the A_APPOINTMENTS record
update the A_APPOINTMENTS record set CCS_CC_CLIENT_ID to null and ATS_ATTEND_STATUS_CODE = ‘P’ and GRO_GROUP_ID = null and WE_WORK_EVENT_ID = null and CCS_ID = null
Delete A_GROUPS record

End loop
End loop

Insert into ENV_VARIABLES (code, env_value, date_created, created_by, date_modified, modified_by, note)
values (C_CLIENTS.CLIENT_ID, term_code, sysdate, USERID, null, null,'termscript');

end if active_client = 'N'

End loop

End

1.3.2.4 Outputs

aas_appointments
aas_appt_items
aas_appt_clients
a_appointments

a_appt_appt_items
a_appt_topic_materials
a_appt_topics
a_groups
c_cert_term_reasons
c_certifications
c_client_groups
c_client_ne_topics
c_client_services
c_client_svc_ne_materials
c_clients
f_wait_lists

1.4 Perform automatic category changes

1.4.1 EA1_CAT_EOD.SQL

Overview

This script performs automatic category changes for clients based upon their certification category codes, current date and birth date. The following table outlines updates that take place on the client certification record. (Does not apply to clients classified in check out clinic status, i.e. o_organizational_units.check_out_flag = 'N'):

Old code	**Months between current month and birth date**	**New code**
IEN, IPN, IFF	12	C1
C1	24	C2
C2	36	C3
C3	48	C4
C4	60	C5
PG1	216	PG2

1.4.1.1 Inputs

c_cat_goals
c_cat_referrals
c_cat_nutr_eds
c_certifications
c_client_goals
c_client_ne_topics
c_client_services
c_clients
c_family_economic_units
c_health_risk_factors
o_organizational_units
c_cert_term_reasons

1.4.1.2 Selection

Select all clients where category code in (‘IEN’,’IPN’,’IFF’,’C1',’C2',’C3',’C4',’PG1') and the clinic the client is assigned to is not checked out and the current date is between the start and end certification dates, and the termination date is null. (This picks up the client’s current certification record.)

1.4.1.3 Processing

For the C_CLIENTS / C_CERTIFICATIONS records:

If C_CERTIFICATIONS.PROGRAM = ‘WIC’

```
    cmp_date := MONTHS_BETWEEN(SYSDATE, C_CLIENTS.BIRTH_DATE)

    if category code starts with ‘I’ and cmp_date >=12
        new_cat_code := ‘C1'
        Run procedure update_cert
    elseif category code = ‘C1' and cmp_date >=24
        new_cat_code := ‘C2'
        Run procedure update_cert
    elseif category code = ‘C2' and cmp_date >= 36
        new_cat_code := ‘C3'
        Run procedure update_cert
    elseif category code = ‘C3' and cmp_date >= 48
        new_cat_code := ‘C4'
        Run procedure update_cert
    elseif category code = ‘PG1' and cmp_date >= 216
        new_cat_code := ‘PG2'
        Run procedure update_cert
    end if
end if
```

If C_CERTIFICATIONS.PROGRAM = ‘CSF’

```
    cmp_date := MONTHS_BETWEEN(SYSDATE, LAST_DAY(C_CLIENTS.BIRTH_DATE))

    if category code starts with ‘I’ and cmp_date >=12
        new_cat_code := ‘C1'
        Run procedure update_cert
    elseif category code = ‘C1' and cmp_date >=24
        new_cat_code := ‘C2'
        Run procedure update_cert
    elseif category code = ‘C2' and cmp_date >= 36
        new_cat_code := ‘C3'
        Run procedure update_cert
    elseif category code = ‘C3' and cmp_date >= 48
        new_cat_code := ‘C4'
```

```
        Run procedure update_cert
    elseif category code = 'C4' and cmp_date >= 60
        new_cat_code := 'C5'
        Run procedure update_cert
    elseif category code = 'PG1' and cmp_date >= 216
        new_cat_code := 'PG2'
        Run procedure update_cert
    end if
end if
```

Procedure **update_cert**:

For infants and children

Update c_certifications set cert_end_date = (birthday-1) , termination date = birthday

For pregnant women

Update c_certifications set cert_end_date = (birthday - 1), termination date = birthday-1,

Update c_food_package_prescriptions set end_date = (birthday-1)

If new_cat_code = 'C1' and priority = 2, then new priority := 3

Set start/end dates for calculation of duration of certification:
calc_start_date := current_date + 1 day
calc_end_date := certification end date (c_end_date)
certification duration := (end date - start date + 1 day) / 7 days
set certification duration to 1 if it is <= 0

Set the new certification end date from the current date plus the new cert period:
Current date + 1 day + (calc_end_date - calc_start_date)

Insert into c_certifications using the calc_start_date, calc_end_date, priority code, new_cat_code, certification duration.

Insert into c_cert_term_reasons for the old certification record, set tr_termination_code = 'Q'

Update c_food_package_prescriptions set cc1_cert_start_date := calc_start_date for all food package prescriptions whose effective_date is after the current date.

Select c_cat_referrals (category referrals) for the client's category code

Duplicate the c_cat_referrals for the client's category into c_client_referrals by inserting into c_client_referrals with c_cat_referrals.prg_program_id and c_cat_referrals.to_from_flag

Select crf_risk_factor_id, crf_cat_category_code, cc1_cc_client_id, cc1_cert_start_date
From c_health_risk_factors for the new certification period.

Insert duplicate records into c_health_risk_factors until no more records are found.

Select c_cat_goals (category goals) for the client's category code

If the client has no goals for that category, duplicate the c_cat_goals for the client's category into c_client_goals with c_cat_goals.g1_goal_code and standard_flag = 'Y'

Select c_client_services for the client with appt_flag = 'N'
    If no records were found, then
        Find next available id value from c_client_services: next_id
        Insert into c_client_services setting key_id = next_id, ats_attend_status_code = 'P', service_date = current date, appt_flag = 'N', cat_category_code to the client's category
    Else
        Delete c_client_ne_topics for the client and most recent and c_client_services.id
    End if

Get all net_nutr_ed_topic_code records for the client's category
Check if the nutrition education topics already exist for the client
    If not, insert into c_client_ne_topics, setting standard_flag = 'Y', received_flag = 'N'
    Else, update the c_client_ne_topics set standard_flag = 'Y'

End Procedure **update_cert**:

1.4.1.4 Outputs

c_certifications
c_client_goals
c_client_ne_topics
c_client_referrals
c_client_services
c_cert_term_reasons
c_food_package_prescriptions
c_health_risk_factors

1.4.2 EA1_CAT_SYNC_EOD.SQL

Overview

Client record's category code is kept in sync with the certification record's category code. In the case of a future certification, the c_clients.cat_category_code maintains the current certification category until the current cert period ends. When the current cert period ends, this eod process will change the c_clients.cat_category_code to equal the new c_certifications.cat_category_code.

1.4.2.1 Inputs

c_clients
c_certifications

1.4.2.2 Selection

Select all client records where category code does not equal the category code of the current unterminated cert record

1.4.2.3 Processing

Main routine

For all client records with a category mismatch, update the c_clients.cat_category_code to the current c_certifications.cat_category_code.

```
update c_clients
set cat_category_code = cat_code,
      modified_by = user,
      date_modified = sysdate
where client_id = cli_id;
```

1.4.2.4 Outputs

c_clients

1.5 Mark appointments as Missed or Kept

1.5.1 EA1_APPT_EOD.SQL

Overview

Client appointment status is updated from pending (P) to kept (K) for client's who's appointment attend status code is 'P' and the calendar date is today or earlier, depending on the following conditions (does not apply to clients classified in check out clinic status):

* If food instruments have been issued for the appointment month
* AND the disposition of any food instrument created on the date of the appointment
* AND/OR medical or bloodwork records create date is greater than or equal to the appointment date

Client appointment status is updated from pending (P) to missed (M) if the above conditions are not met.

If the participant is a mother of an exclusively breastfeeding infant, and she meets the criterion in 1-3 to have her appointment updated to kept (K) then the infant's appointment record will be updated to kept (K) also.

1.5.1.1 Inputs

a_appointments
c_bloodwork_data
c_client_groups
c_infant_child_medicals
c_woman_medicals
i_food_instruments

1.5.1.2 Selection

Client appointments:
Select all a_appointments records where ats_attend_status_code = 'P' and cd_calendar_date is today or earlier.

Group appointments:
Select all a_appointments records where ats_attend_status_code = 'P' and cd_calendar_date is today or earlier.

1.5.1.3 Processing

Main routine

1. for all pending client appointments today or earlier, if attend_code = 'P',
   Run procedure **check_for_x**:

   Search for all food instruments that has issue_date in the current month, and idis_disposition_code = '2'; if found, set found_it := 'Y'
   Search for c_bloodwork_data that has the date_created equal to the appointment calendar date; if found, set found_it := 'Y'
   Search for c_infant_child_medicals that has date_created equal to the appointment calendar date; if found, set found_it := 'Y'
   Search for c_woman_medicals that has date_created equal to the appointment calendar date; if found, set found_it := 'Y'

   If found_it = 'N',
   ats_code := 'M'
   Else
   ats_code := 'K'

   Run procedure **update_appointments**:

   Update a_appointments set ats_attend_status_code := ats_code

   Run procedure **update_client_services**:

   Update c_client_services set ats_atten_status_code := ats_code for all pending client appointments today or earlier, if attend_code = 'P',

2. For all pending group appointments today or earlier if attend_code = 'P' for clients in the group, set client services and groups to kept/missed.

   For all clients in the group,

   Run procedure **check_for_x**:

   Search for all food instruments that has issue_date in the current month, and idis_disposition_code = '2'; if found, set found_it := 'Y'
   Search for c_bloodwork_data that has the date_created equal to the appointment calendar date; if found, set found_it := 'Y'
   Search for c_infant_child_medicals that has date_created equal to the appointment calendar date; if found, set found_it := 'Y'
   Search for c_woman_medicals that has date_created equal to the appointment calendar date; if found, set found_it := 'Y'

   If found_it = 'N',

ats_code := ‘M’
Else
ats_code := ‘K’

Update c_client_services set ats_atten_status_code := ats_code
Update c_client_groups set ats_attend_status_code := ats_code

2a. And for the same group appointments: update appointments to missed if all clients in the group are missed.

Set temp_ats_code to ‘A’
if ats_attend_status_code = ‘K’ on the c_client_groups table for the group,
Set temp_ats_code to ‘Z’

If temp_ats_code = ‘Z’,
Update a_appointments (the group appointment) set ats_attend_status_code = ‘K’
Else
Update a_appointments (the group appointment) set ats_attend_status_code = ‘M’

If the previous processing updates the participants attend_status_code to 'K' and the participant is the mother of an exclusively breastfeeding infant, c_clients.cat_category_code = 'IEN' and c_clients.cc_client_id = c_clients.client_id (the mother's client id) and the mother's a_appointments.acd_calendar_day and a_appointments.start_time equal the infant's a_appointments.acd_calendar_day and a_appointments.start_time the infant's appointment record will be updated to kept (K) also.

1.5.1.4 Outputs

a_appoinments
c_client_services
c_client_groups

1.6 Delete information no longer used by the system

1.6.1 EA1_OTHER_PURG.SQL

Overview

This script deletes records that are no longer used by the system. The following outlines records that are deleted and any conditions relating to the deletion of these records (does not apply to clients classified in check out clinic status):

* Client records will be deleted, as well as their child records, if the client has no certification record or the client record has no appointment date and have not been waitlisted .
* Client records will be deleted, as well as their child records, with no application date and date created/date modified more than 60 days ago who have not been waitlisted
* The family records will be deleted, as well as the child records, if the family records have no related client records.
* The food instrument records will be deleted if the issue date is greater than twenty four months ago.
* The del statements records whose date created is earlier than the first day of two months ago.
* Transfer requests that are more than 1 month old.
* Income history records that do not have any dependent income records.
* Leftover records in the temporary food instrument table.
* Archived client records that were created more than 15 days in the past.

1.6.1.1 Inputs

a_appointments
a_appt_appt_items
a_appt_topic_materials
a_appt_topics
c_b_n_hh_reasons_bfends
c_bloodwork_data
c_cert_peer_counsels
c_cert_term_reasons
c_certifications
c_client_communications
c_client_goals
c_client_groups
c_client_ne_topics
c_client_notes
c_client_progs
c_client_referrals
c_client_services
c_client_svc_ne_materials
c_clients

c_contacts
c_dietary_assessments
c_economic_unit_members
c_family_economic_units
c_family_economic_units cfeu,
c_family_phones
c_feu_communications
c_food_package_prescriptions
c_health_risk_factors
c_i_c_healths
c_i_c_hh_reasons_bfends
c_income_histories
c_incomes
c_infant_child_medicals
c_infant_data
c_infant_data
c_p_couns_notes
c_peer_rbfends
c_resolutions
c_transfer_comms
c_transfer_histories
c_transfer_phones
c_transfers_info
c_woman_medicals
del_statements
f_wait_list_contacts
f_wait_lists
i_bf_promo_vchrs
i_fi_reject_reasons
i_food_instruments
i_temp_fis
o_organizational_units
r_archived_clients
v_compliance_case_clients

1.6.1.2 Selection

1. clients without a certification record, created more than 120 days ago

2. clients with no application date and date created/modified more than 120 days ago

3. c_family_economic_units with no client records

4. i_food_instruments records whose issue date is more than 24 months ago

5. c_transfers_info records with date_created more than ten days 1 month in the past

6. del_statements records whose date_created is earler than the first day of two months ago

7. c_income_histories records without a dependent c_income record

8. Any i_temp_fis records remaining

9. r_archived clients records with date_created more than 15 days in the past.

1.6.1.3 Processing

1. For all clients without a certification record, created more than 120 days ago, and not checked out:

Run procedure **delete_client_children**:

Procedure **delete_client_children**

Clid := C_CLIENTS.CLIENT_ID

Delete the following records for the client:

For all appointments for the client, delete:
a_appt_topic_materials
a_appt_topics
a_appt_appt_items
Update a_appointments set ats_attend_status_code = 'P', css_cc_client_id = null, css_id = null

c_client_svc_ne_materials
c_prev_names
c_more_ethnic_groups
c_prev_families
c_bf_promo_issuances
cw_reasons_bfends
i_food_instruments
f_wait_list_contacts
c_diet_nutrients
c_i_c_hh_reasons_bfends
c_client_ne_topics
c_woman_medicals
c_bloodwork_data
c_cert_term_reasons
c_b_n_healths
c_food_package_prescriptions
f_wait_lists
c_p_healths
c_infant_child_medicals
c_dietary_assessments
c_health_risk_factors
c_i_c_healths
c_peer_rbfends

c_contacts
c_p_couns_notes
c_resolved_clients
i_bf_promo_vchrs
v_compliance_case_clients
c_client_services
c_certifications
c_transfer_histories
c_immunizations
c_client_referrals
c_client_communications
c_resolutions
c_client_notes
c_client_progs
c_client_groups
c_infant_data
c_client_goals
c_cert_peer_counsels

Delete from C_INFANT_DATA where cc_client_id_mother = clid;

Update c_clients set cc_client_id = null

Loop through all c_income_histories records for the client and
Delete all related c_incomes, c_economic_unit_members,
c_income_histories

Delete from C_CLIENTS where client_id=clid;

End Procedure **delete_client_children**

2. For all clients with no application date and date created/modified more than 120 days ago
   Run procedure **delete_client_children**: (see above)

3. For all c_family_economic_units with no client records:

   Delete all c_income records where family has a c_income_histories record.
   Delete all c_economic_unit_members where family has a c_income_histories record.
   Delete the following records for the family:
   - c_income_histories
   - c_feu_communications
   - c_family_phones
   - c_family_economic_units
   - c_fam_referrals
   - c_smoking_histories
   - c_economic_unit_members
   - c_incomes

4. For all i_food_instruments records whose issue date is more than 24 months ago,

   Delete the following record:
   - i_fi_reject_reasons

i_food_instruments

5. For all c_transfers_info records with date_created more than ten days 1 month in the past,

   Delete the following records for the transfer:
   c_transfer_comms
   c_transfer_phones
   c_transfers_info

6. Delete all del_statements whose date_created is earlier than the first day of two months ago.

7. Delete all c_income_histories_records that do not have dependent c_income records. This is to purge extra income history records created by the income calculator.

8. Delete any records remaining in the i_temp_fis table. If food instrument printing was not successful during the business day, unwanted records may be left in this table. So they need to be purged.

9. Delete any r_archived_clients records with a date_created more than 15 days in the past.

1.6.1.4 Outputs

a_appointments
a_appt_appt_items
a_appt_topic_materials
a_appt_topics
c_b_n_healths
c_b_n_hh_reasons_bfends
c_bloodwork_data
c_cert_peer_counsels
c_cert_term_reasons
c_certifications
c_client_communications
c_client_goals
c_client_groups
c_client_ne_topics
c_client_notes
c_client_progs
c_client_referrals
c_client_services
c_client_svc_ne_materials
c_clients
c_contacts
c_diet_nutrients
c_dietary_assessments
c_economic_unit_members
c_family_economic_units
c_family_economic_units
c_family_phones
c_feu_communications

c_food_package_prescriptions
c_health_risk_factors
c_i_c_healths
c_i_c_hh_reasons_bfends
c_immunizations
c_income_histories
c_income_histories
c_incomes
c_infant_child_medicals
c_infant_data
c_p_couns_notes
c_p_healths
c_peer_rbfends
c_resolutions
c_transfer_comms
c_transfer_histories
c_transfer_phones
c_transfers_info
c_woman_medicals
del_statements
f_wait_list_contacts
f_wait_lists
i_bf_promo_vchrs
i_fi_reject_reasons
i_food_instruments
i_temp_fis
o_organizational_units
r_archived_clients
v_compliance_case_clients
c_prev_names
c_prev_families
c_more_ethnic_groups
c_bf_promo_issuances
c_smoking_histories
c_economic_unit_members
c_incomes

1.7 Set exclusively breastfeeding clients to active status

1.7.1 EA1_IEN_ACTIVE.SQL

Overview

Note: Currently, Arizona WIC is not using this form.

Clients which are exclusively breastfeeding have their records updated to a status of active (A) from a record status of pending (P). The category code that makes a client exclusively breastfeeding is a category of 'IEN' (does not apply to inactive infants in check out clinic status).

1.7.1.1 Inputs

c_bloodwork_data
c_certifications
c_clients
c_family_economic_units
c_food_package_prescriptions
c_i_c_healths
c_infant_child_medicals
eod_controls
o_organizational_units

1.7.1.2 Selection

For all c_clients with cat_category_code = 'IEN' (breastfeeding infant) and rec_status = 'P' (pending)

1.7.1.3 Processing

For all c_clients with cat_category_code = 'IEN' (breastfeeding infant) and rec_status = 'P' (pending)
Get the current certification record: cert_start_date. For the infant's current certification:
Get c_i_c_healths records: v_found := count(*)
If v_found > 0 then
Get c_infant_child_medicals record: v_found := count(*)
If v_found > 0 and birth date is more than six months ago
Get c_bloodwork_data record: v_found := count(*)

If a current certification record found and v_found > 0
Get c_food_package_prescriptions for the infant's current certification: v_found := count(*)

If v_found = 0 (no food package prescriptions were found for the infant's current certification period:Update c_clients set rec_status = 'A', date_modified = eod date

1.7.1.4 Outputs

c_clients

1.8 Process any pending transfer requests

1.8.1 EA1_TRANSFER.SQL

Overview

Any transfer requests that could not be completed during the day will be performed at this time.

1.8.1.1 Inputs

none

1.8.1.2 Selection

none

1.8.1.3 Processing

Runs the transfer_pkg.initiate_transfer database procedure which loops through the pending transfer requests in c_transfers_info while processing each one.

1.8.1.4 Outputs

Prints message (hold_message) to agcy_sql.log indicating transfer_ids that were processed including the number of successful and unsuccessful transfers.

Ex.
Transfer ID: 10013
Transfers Successful: 1 Transfers Failed: 0

1.9 Generate notices, letters, and reports

1.9.1 CS_LETTER_OF_TERM.FMX

Overview

This form prints out the ineligibility notices for participants who have been terminated by ea1_term_eod and ea1_term_wl_clients. This form also prints appointment notices 14 days before the participant's appointment. A notice of Ineligibility is printed six weeks prior to the end of the certification period for Participants that will no longer be categorically eligible for CSF. The system also prints the CSFP Certifications Due report and a Notice to Reapply for CSF Participants six weeks prior to the end of their certification period if they remain potentially categorically eligible. This form also prints the Pending Food Package Approval report listing all participants with food packages that require approval.

The system prints the reports and notices for participants at their assigned clinic. This will be facilitated through the identification of the clinic printers for each Local Agency. Each printer will have a unique "name" that the Local Agency/Clinic server will use to initiate print commands over the Arizona WIC WAN. As of this submission, the new Server/Printers have not been implemented, so a list of the Clinic printer names cannot be provided.

1.9.1.1 Inputs

a_attend_status
a_appointments
c_clients
c_certifications
c_client_services

1.9.1.2 Selection

Ineligibility Notice

C_CERTIFICATIONS where termination_date is null and sysdate equals max(c_certifications.cert_end_date) minus six weeks and a C_CLIENT_COMMUNICATIONS record does not exist where (date_sent_called is >= c_certifications.cert_end_date - 6 weeks and ct_comm_type_code = 'IN') and any of the following conditions are true:

1. c_certifications.cat_category_code in 'PG1','PG2'(Pregnant) and the cert_end_date is 42 days after the last day of the month for the c_certifications.expected_delivery_date
2. c_certifications.cat_category_code = 'C5'(child) and the cert_end_date is the last day of the 72nd month after c_clients.birth_date
3. c_certifications.cat_category_code in 'P', 'EN', 'PN'(Postpartum or Breastfeeding) and the cert_end_date is the last day of the 12th month after c_certifications.actual_delivery_date.

Notice to Reapply for CSF Participants

C_CERTIFICATIONS where termination_date is null and sysdate equals max(c_certifications.cert_end_date) minus six weeks and a C_CLIENT_COMMUNICATIONS record does not exist where (date_sent_called is >= c_certifications.cert_end_date - 6 weeks and ct_comm_type_code = 'RN') and any of the following conditions are true:

1. c_certifications.cat_category_code in 'E1', 'E2', 'E3', 'E4' (Elderly).
2. c_certifications.cat_category_code = 'C5'(child) and the cert_end_date is less than the last day of the 72nd month after c_clients.birth_date
3. c_certifications.cat_category_code in 'P', 'EN', 'PN'(Postpartum or Breastfeeding) and the cert_end_date is less than the last day of the 12th month after c_certifications.actual_delivery_date.

CSF Recerts Due Report

Same criteria as the Notice to Reapply selection.

Appointment Notices

C_CLIENTS where sysdate equals the a_appointments.cd_calendar_date minus 14 days for the c_client.

The O_ORGANIZATIONAL_UNITS.PRINT_APPT_NOTICES must be checked on the O_ORGANIZATIONAL_UNITS screen in Operations Management in order to print the appointment notices during End of Day.

Pending Food Package Approval

C_CLIENTS where C_FOOD_PACKAGE_PRESCRIPTION.REQUIRES_APPROVAL = 'Y' and C_FOOD_PACKAGE_PRESCRIPTION.NUTRITIONIST_FLAG and PHYSICIAN_FLAG = 'N' and C_CERTIFICATIONS.TERMINATION_DATE is null.

1.9.1.3 Processing

Ineligibility Notice

For each C_CERTIFICATIONS record found meeting the above criteria
Loop
    if C_FAMILY_ECONOMIC_UNITS.LANG_LANGUAGE_CODE = '2' (Spanish)
        Print the MS Word document cs_letter_of_inelig_sp.doc
    else
        Print the MS Word document cs_letter_of_inelig.doc.
    end if
End loop

Notice to Reapply for CSF Participants

For each C_CERTIFICATIONS record found meeting the above criteria
Loop
    if C_FAMILY_ECONOMIC_UNITS.LANG_LANGUAGE_CODE = '2' (Spanish)
        Print the MS Word document cs_csf_reapply_notice_sp.doc
    else
        Print the MS Word document cs_csf_reapply_notice.doc
    end if
End loop

CSFP Certifications Due Report

Invoke and print the Oracle Report file cr_csf_recerts_due

Appointment Notices

For each C_CLIENTS record found meeting the above criteria
Loop

Print the MS Word document appt_notice.doc

End Loop

Pending Food Package Approval

Invoke and print the Oracle report file cr_pend_fp_appr

1.9.1.4 Outputs

Reference the following sections of the DTSD for layouts of the notices and reports:

Section 2 - 4.1.10	Notice to Reapply
Section 2 - 4.1.4	Notice of Ineligibility
Section 2- 4.3.37	CSFP Certifications Due
Section 1-1.1.1.10	Notice of Appointment.
Section 2- 4.3.45	Pending Food Package Approval Report

1.10 Gather updated client information, new issuance information, void information, and requests for transfer

1.10.1 EA1_TAB_UP.SQL

Overview

This script extracts client information that has been changed or created. It compares the change or created date in each of the following tables to the last day that End of Day Processes ran successfully.

- Staff members
- Staff phones
- Staff time studies
- Job descriptions
- Food instruments
- Food instrument types
- Food instrument foods
- Food instrument formulas
- Food packages
- Food package food instrument types
- Food package foods
- Food package prescriptions
- FI formulas
- Package distributions
- Organization units
- Organization programs
- Organization unit phones
- Outreach organizations
- Outreach comms
- Outreach organizations phones
- Outreach programs
- Outreach Org LAs
- Family economic units
- Family phones
- Family referrals
- FEU communications
- Clients
- Clients BP Issuances
- Client communications
- Client goals
- Client notes
- Client programs
- Client referrals
- Client services

- Client more ethnic groups
- Client NE topics
- Client services NE materials
- Client previous names
- Certified peer counsels
- Counsel notes
- Client allergy foods
- Peer Rbfends
- Contacts
- Income histories
- Economic unit members
- Incomes
- Immunizations
- Infant data
- Infant child medicals
- Resolutions
- Resolved clients
- Transfer information
- Transfer histories
- Certifications
- Certification term reasons
- Cert nutr eds
- Bloodwork data
- Dietary assessments
- Diet nutrients
- Health risk factors
- C Food Package Prescriptions
- C B W healths
- C B W HH reasons bfends
- C I C healths
- C I C HH reasons bfends
- C P healths
- C P S responses
- C food box distributions
- C infant child medicals
- C woman medicals
- F case assignments
- F caseload restrictions
- F wait lists
- F wait list contacts
- I BF promotional vouchers
- I BF voucher items
- I FI inventories
- I FI ranges
- I inventories
- I inventory organizational units
- I stock inventories
- I inv org units

- V complaints
- V monthly activities
- V activity findings
- Delete statements
- Batch runs
- Printer Addresses

1.10.1.1 Inputs

EOD_CONTROLS
O_STAFF_MEMBERS
O_STAFF_TIMSTUDIES
I_FOOD_INSTRUMENT_TYPES
O_ORGANIZATIONAL_UNITS
I_FOOD_INSTRUMENT_FOODS
O_ORG_PROGRAMS
I_FOOD_INSTRUMENTS
O_ORG_UNIT_PHONES
I_FI_FORMULAS
O_OUTREACH_ORGANIZATIONS
O_OUT_ORG_LAS
I_FOOD_PACKAGFI_TYPES
O_OUTREACH_COMMS
O_OUTREACH_ORG_PHONES
O_OUTREACH_PROGRAMS
O_STAFF_PHONES
O_JOB_DESCRIPTIONS
C_FAMILY_ECONOMIC_UNITS
C_FAMILY_PHONES
C_FAM_REFERRALS
C_FEU_COMMUNICATIONS
C_SMOKING_HISTORIES
C_PREV_FAMILIES
C_CLIENTS
C_CERT_PEER_COUNSELS
C_CONTACTS
C_PEER_RBFENDS
C_P_COUNS_NOTES
C_CLIENT_COMMUNICATIONS
C_CLIENT_GOALS
C_CLIENT_NOTES
C_CLIENT_PROGS
C_CLIENT_REFERRALS
C_CLIENT_SERVICES
C_CLIENT_NTOPICS
C_CLIENT_SVC_NMATERIALS
C_MORETHNIC_GROUPS
C_PREV_NAMES
C_BF_PROMO_ISSUANCES

C_ALLERGY_FOODS
C_W_HEALTHS
C_W_HH_REASONS_BFENDS
C_PS_RESPONSES
C_CLIENT_BP_ISSUANCES
C_INCOMHISTORIES
C_ECONOMIC_UNIT_MEMBERS
C_INCOMES
C_INFANT_DATA
C_RESOLUTIONS
C_RESOLVED_CLIENTS
C_TRANSFERS_INFO
C_TRANSFER_HISTORIES
C_CERTIFICATIONS
C_CERT_NUTR_EDS
C_FOOD_BOX_DISTS
C_BLOODWORK_DATA
C_CERT_TERM_REASONS
C_DIETARY_ASSESSMENTS
C_DIET_NUTRIENTS
C_HEALTH_RISK_FACTORS
C_INFANT_CHILD_MEDICALS
C_WOMAN_MEDICALS
C_I_C_HEALTHS
C_I_C_HH_REASONS_BFENDS
F_CASASSIGNMENTS
F_WAIT_LISTS
F_WAIT_LIST_CONTACTS
F_CASELOAD_RESTRICTIONS
I_FOOD_PACKAGES
I_FOOD_PACKAGFOODS
I_PACKAGDISTRIBUTIONS
C_FOOD_PACKAGPRESCRIPTIONS
I_BF_PROMO_VCHRS
I_BF_VCHR_ITEMS
I_FI_INVENTORIES
I_INVENTORIES
I_STOCK_INVENTORIES
I_INV_ORG_UNITS
V_COMPLAINTS
V_MON_ACTIVITIES
V_ACTIVITY_FINDINGS
I_BATCH_RUNS
S_PRINTER_ADDRESSES
DEL_STATEMENTS

1.10.1.2 Selection

Only update records that have changed since the last successful end of day process:

All of the input tables WHERE (NVL(date_modified, date_created) > (SELECT good_proc_date FROM eod_controls))
Date_modified or date_created is greater than good_proc_date from eod_controls table.

1.10.1.3 Processing

1. Each record in all of the input tables matching the selection criteria are inserted into the corresponding output table.

2. INSERT INTO EODADM.E_EOD_CONTROLS(TO_DATE, FROM_DATE, GOOD_PROC_DATE)

1.10.1.4 Outputs

E_EOD_CONTROLS
E_O_STAFF_MEMBERS
E_O_STAFF_TIME_STUDIES
E_I_FOOD_INSTRUMENT_TYPES
E_O_ORGANIZATIONAL_UNITS
E_I_FOOD_INSTRUMENT_FOODS
E_O_ORG_PROGRAMS
E_I_FOOD_INSTRUMENTS
E_O_ORG_UNIT_PHONES
E_I_FI_FORMULAS
E_O_OUTREACH_ORGANIZATIONS
E_O_OUT_ORG_LAS
E_I_FOOD_PACKAGE_FI_TYPES
E_O_OUTREACH_COMMS
E_O_OUTREACH_ORG_PHONES
E_O_OUTREACH_PROGRAMS
E_O_STAFF_PHONES
E_O_JOB_DESCRIPTIONS
E_C_FAMILY_ECONOMIC_UNITS
E_C_FAMILY_PHONES
E_C_FAM_REFERRALS
E_C_FEU_COMMUNICATIONS
E_C_SMOKING_HISTORIES
E_C_PREV_FAMILIES
E_C_CLIENTS
E_C_CERT_PEER_COUNSELS
E_C_CONTACTS
E_C_PEER_RBFENDS
E_C_P_COUNS_NOTES
E_C_CLIENT_COMMUNICATIONS
E_C_CLIENT_GOALS
E_C_CLIENT_NOTES
E_C_CLIENT_PROGS
E_C_CLIENT_REFERRALS
E_C_CLIENT_SERVICES
E_C_CLIENT_NE_TOPICS
E_C_CLIENT_SVC_NE_MATERIALS
E_C_MORE_ETHNIC_GROUPS
E_C_PREV_NAMES
E_C_BF_PROMO_ISSUANCES
E_C_ALLERGY_FOODS
E_C_W_HEALTHS
E_C_W_HH_REASONS_BFENDS
E_C_PS_RESPONSES
E_C_CLIENT_BP_ISSUANCES
E_C_INCOME_HISTORIES

E_C_ECONOMIC_UNIT_MEMBERS
E_C_INCOMES
E_C_INFANT_DATA
E_C_RESOLUTIONS
E_C_RESOLVED_CLIENTS
E_C_TRANSFERS_INFO
E_C_TRANSFER_HISTORIES
E_C_CERTIFICATIONS
E_C_CERT_NUTR_EDS
E_C_FOOD_BOX_DISTS
E_C_BLOODWORK_DATA
E_C_CERT_TERM_REASONS
E_C_DIETARY_ASSESSMENTS
E_C_DIET_NUTRIENTS
E_C_HEALTH_RISK_FACTORS
E_C_INFANT_CHILD_MEDICALS
E_C_WOMAN_MEDICALS
E_C_I_C_HEALTHS
E_C_I_C_HH_REASONS_BFENDS
E_F_CASE_ASSIGNMENTS
E_F_WAIT_LISTS
E_F_WAIT_LIST_CONTACTS
E_F_CASELOAD_RESTRICTIONS
E_I_FOOD_PACKAGES
E_I_FOOD_PACKAGE_FOODS
E_I_PACKAGE_DISTRIBUTIONS
E_C_FOOD_PACKAGE_PRESCRIPTIONS
E_I_BF_PROMO_VCHRS
E_I_BF_VCHR_ITEMS
E_I_FI_INVENTORIES
E_I_INVENTORIES
E_I_STOCK_INVENTORIES
E_I_INV_ORG_UNITS
E_V_COMPLAINTS
E_V_MON_ACTIVITIES
E_V_ACTIVITY_FINDINGS
E_I_BATCH_RUNS
E_S_PRINTER_ADDRESSES
E_DEL_STATEMENTS

1.10.2 EA1_EXPORT_TAB.TXT

Overview

This is a text file containing the parameters used by the Oracle Export command that is executed by the EA1_EXPORT.BAT batch file. The output tables listed above in section 1.10.1.4 are listed in this parameter file and are exported to a file that is later copied to the central server to be imported into the central database.

1.10.2.1 Inputs

N/A

1.10.2.2 Selection

N/A

1.10.2.3 Processing

N/A

1.10.2.4 Outputs

N/A

1.11 Initiate Polling Process

1.11.1 EC1.BAT

Overview

This batch file begins EOD processing at Central.

Processing

This script is called from the Systems Administration End of Day form when the proceed button is pressed and the AGCY_SEQ is equal to ‘00'.

The script reads in the EC1.INI file to initialize and set the temporary environmental variables.

The script begins by running the SQL script EC1.SQL which updates EOD_CONTROLS and ENV_VARIABLES and then runs the following SQL scripts:

EC1_OTHER_PURG.SQL
EC1_PRG_ARCHV.SQL
EC1_CTRL_CHCK.SQL
EC1_STALE_DATE_FI.SQL

This it calls the following scripts to prepare the vendor PGA bank files.

EC1_OUTBOUND_PGA_FSMC.SQL
EC1_PGA_FSMC.SQL
EC1_OUTBOUND_VENDOR_FSMC.SQL
EC1_VENDOR_FSMC.SQL
EC1_VENDOR_FSMC.BAT

This script then calls EC2.BAT, EC4.BAT, and EC5.BAT to complete the EOD process at central.

1.12 Delete information no longer used by the system

1.12.1 EC1.SQL

Overview

Executes SQL scripts for EC1.BAT.

1.12.1.1 Inputs

eod_controls
env_variables

1.12.1.2 Selection

env_variables
where code = ‘EOD_BY’

1.12.1.3 Processing

update eod_controls, setting TO_DATE to the system date
update env_variables, setting env_value to ‘EC1' where code = ‘EOD_BY’

run SQL script EC1_OTHER_PURG
run SQL script EC1_PRG_ARCHV
run SQL script EC1_CTRL_CHCK
run SQL script EC1_STALE_DATE_FI

1.12.1.4 Outputs

end_controls
env_varibles

Log files: EC1_SQL.LOG

1.12.2 EC1_OTHER_PURG.SQL

Overview

This script deletes various records that are no longer used by the system. The following records are removed if conditions are met (does not apply to clients classified in check out clinic status):

Records Deleted	Conditions
Delete statements	Del-statements with date_created more than 2 months in the past
Income history records	Delete records that don't have corresponding records in c_incomes table
Temp FI records	Delete any records remaining in i_temp_fis because of printing problems
Archived/client records	Delete r_archived_clients records with a date_created more than 15 days in the past

1.12.2.1 Inputs

DEL_STATEMENTS

1.12.2.2 Selection

Determines which records are no longer used by the system. This is done in the following processing step in the where clause.

1.12.2.3 Processing

Deletes unused records:

DELETE FROM DEL_STATEMENTS
WHERE date_created < min_date (min date is two months prior to the current date).

DELETE FROM C_ECONOMIC_UNIT_MEMBERS
WHERE cih_income_history_id = inc_rec.income_history_id (inc_rec.income_history_id is any income_history_id that does not have corresponding c_income records)

DELETE from C_INCOME_HISTORIES
WHERE income_history_id=inc_rec.income_history_id

DELETE from I_TEMP_FIS

DELETE from R_ARCHIVED_CLIENTS
WHERE NVL (date_modified, date_created) < (sysdate - 15)

1.12.2.4 Outputs

del_statements

1.12.3 EC1_STALE_DATE_FI.SQL

Overview

Updates i_food_instruments for issued food instruments which have passed their stale date and are no longer redeemable.

1.12.3.1 Input

N/A

1.12.3.2 Selection

i_food_instruments
where idis_disposition_code is 1 or 2 (printed not issued, issued) or 5 (rejected)
and void_date is null (not voided)
and cleared_date is null (not redeemed)
and stale_date is before the system date (stale date in the past)

1.12.3.3 Processing

Select the serial_number for i_food_instruments records meeting the selection criteria

Loop through the i_food_instruments records with the serial_number values:

update i_food_instruments, setting
ivr_void_reason_code = 'Z' (stale),
idis_disposition_code = '3' (voided),
void_date = current date
date_modified – Sysdate
voided_by – 'EODADM'

Insert the i_food_instrument into i_bank_reports, setting:
post_code = 'Y'
send_code = 'B'
message = 'STALE DATE WITH NORMAL POST'
rec_status = 'U'

end;

1.12.3.4 Outputs

i_bank_reports
i_food_instruments

1.13 Compile direct payment FI information and send to bank

1.13.1 EC1_CTRL_CHCK.SQL

Overview

This script initializes the I_BANK_REPORTS table and creates I_BANK_REPORTS records for food instruments created at the state for compliance cases and replacement checks (issue method 'R').

1.13.1.1 Inputs

I_BANK_REPORTS
I_FOOD_INSTRUMENTS
EOD_CONTROLS

1.13.1.2 Selection

Selection Criteria 1:

Determines which bank report records need to be created from food instruments and sets the record status to "P" (Insert) to indicate the food instrument is new:

```
FROM  i_food_instruments
WHERE (compliance_buy_flag = 'Y'
          OR issue_method = 'P')
         AND date_created >
             (SELECT from_date
              FROM   eod_controls)
         AND cleared date is NULL
```

Selection Criteria 2:

Determines which bank report records need to be created from food instruments that were updated and sets the record status to "P" to indicate the food instrument was updated:

```
FROM  i_food_instruments
WHERE  date_modified >
             (SELECT from_date
              FROM   eod_controls)
       AND cleared_date is null
       AND (compliance_buy_flag = 'Y' or issue_method = 'P')
```

Selection Criteria 3:

Determines the bank reports that need to be created from food instruments that were revalidated and sets the record status to ‘P’ to indicate the food instrument was updated.

FROM I_food_instruments
WHERE revalidation_code = ‘2’
AND cleared_date is null
AND date_modified >
(select from_date from eod_control)

1.13.1.3 Processing

The scripts performs the following delete to initialize the i_bank_reports table prior to the insert:
DELETE FROM i_bank_reports

Then the following insert is performed based upon selection criteria 1 above:

INSERT INTO i_bank_reports, setting post_code to ‘Y’, send_code to ‘B’, message to ‘POSTED COMPLIANCE/REPLACEMENT DRAFT’, and rec_status to ‘P’.

Then the following insert is performed based upon selection criteria 2 above
INSERT INTO i_bank_reports, setting post_code to ‘Y’, send_code to ‘B’, message to ‘DUPLICATE COMPLIANCE/REPLACEMENT DRAFT’, update no post necessary, and rec_status to ‘P’.

The following is performed based upon selection criteria 3:

INSERT INTO i_bank_reports, setting post_code to ‘Y’, send_code to ‘B’, manage to ‘NORMAL POSTING OF REVALIDATION CHECK’ and’ rec_status to ‘P’.

1.13.1.4 Outputs

I_Bank_Reports

1.14 Compile vendor information to send to bank

1.14.1 EC1_VENDOR_FSMC.BAT

Overview

This batch process creates a vendor text file that will be sent to the bank. If the text file already exists then it will delete it. If the file doesn't exist it will be created. The file is deleted at the end of the batch process. Creates a file of Vendor data to be sent to FSMC: ANVENDOR.DAT.

1.14.1.1 Inputs

vendor_fsmc.dat (see 1.14.2 for file format)

1.14.1.2 Selection

N/A

1.14.1.3 Processing

Creates a vendor text file that will be sent to the bank. If the previous file has not been sent then the new data is appended to the old file.

if the file ANVENDOR.DAT exists,

    Delete VENDOR_FSMC.DAT file
else
    copy the file VENDOR.FSMC.DAT to ANVENDOR.DAT
end;

delete the file VENDOR_FSMC.DAT

1.14.1.4 Outputs

anvendor_fsmc.dat (see 1.14.2 for file format)

1.14.2 EC1_VENDOR_FSMC.SQL

Overview

This script creates a temporary vendor data set to be used for bank processing. The temporary data set is created based upon all records in the vendor table. Creation or change to a record is indicated by the status code of the record.

1.14.2.1 Inputs

V_VENDORS
V_AUTHORIZATIONS
S_GEO_LOCATIONS
V_VENDOR_ACCOUNTS
V_BANK_BRANCHES
V-DISQUALIFICATIONS

1.14.2.2 Selection

A record is created for a Vendor if the following conditions are met:
the v_vendors.vendor_id is not null (the Vendor has been given a Vendor ID),
the Vendor's status does not indicate "Authorization in Progress".

Select Vendors where:

```
vendor_id is not null
and vendors.vs_status_code != 5
and vendors.id = v_authorizations.ven_id
and vendors.sgeo_geo_location_id_mailed = s_geo_locations.geo_location_id
and vendors.id = v_vendor_accounts.ven_id
and v_vendor_accounts.va_vb_routing_number = v_bank_branches.routing_number
and v_vendor_accounts.wic_checks_deposited_flag = 'y'
and v_authorizations.start_date = (select max(start_date)
                                   from v_authorizations
                                   where v_vendor_accounts.ven_id = ven_id)
```

1.14.2.3 Processing

Creates a temporary vendor data set to be used for bank processing. This includes the vendor id, name, peer group, status, mailing address, and bank information.

The detail information includes the following:

```
SELECT DISTINCT(LPAD(ven.vendor_id,5,0))
   ,RPAD(NVL(ven.vendor_name,' '),35,' ')
```

```
  ,RPAD(NVL(SUBSTR(ven.mailing_address1,1,35),' '),35,' ')
  ,RPAD(NVL(SUBSTR(ven.mailing_address2,1,35),' '),35,' ')
  ,RPAD(NVL(sgeo.sc_description,' '),35,' ')
  ,RPAD(NVL(sgeo.ss_state_id,' '),2,' ')
  ,RPAD(NVL(sgeo.sz_zip5||ven.mailing_zip4,0),9,0)
  '    '
  ,LPAD(TO_NUMBER(ven.pee_peer_group_id),2,0)
  ,DECODE(vauth.termination_date,NULL,
    DECODE(vd.start_date,NULL,'00000000',
       DECODE(SIGN(vd.end_date - SYSDATE),-1,'00000000',
       RPAD(TO_CHAR(NVL(vd.start_date,SYSDATE),'mmddyyyy'),8,'0'))),
    DECODE(SIGN(vauth.termination_date - SYSDATE),+1,'00000000',
    RPAD(TO_CHAR(NVL(vauth.termination_date,SYSDATE),'mmddyyyy'),8,'0')))
  ,RPAD(NVL(SUBSTR(vb.bank_name,1,30),' '),30,' ')
  ,LPAD(vb.routing_number,9,0)
  ,RPAD(NVL(va.account_number,' '),10,' ')
```

Also sends another line to the temporary data set, containing the total count of records, preceded by the word “TOTAL”.

The Summary information includes the following:

```
  'TOTAL',
  lpad(ltrim(to_char(count(*))),6,'0')
```

1.14.2.4 Outputs

vendor_fsmc.dat file

Note: For file layout properties, please refer to Appendix B

1.14.3 EC1_PGA_FSMC.BAT

Overview

This script reates a peer group averages data file that will be sent to the bank. If the text file already exists then it will delete it. Otherwise it will create and delte the text file. It creates a file of PGA data to be sent to FSMC: azpgaver.dat.

1.14.3.1 Inputs

p_g_avgs_fsmc.dat

1.14.3.2 Selection

N/A

1.14.3.3 Processing

Creates a PGA text file that will be sent to the bank.

```
if p_g_avgs_fsmc.dat exists,
        delete p_g_avgs_fsmc.dat
else
        copy the p_g_avgs_fsmc.dat to azpgaver.dat
and;
delete p_g_avgs_fsmc.dat
```

1.14.3.4 Outputs

azpgaver.dat

```
SELECT DISTINCT(LPAD(ven.vendor_id,5,0))
   ,RPAD(NVL(ven.vendor_name,' '),35,' ')
   ,RPAD(NVL(SUBSTR(ven.mailing_address1,1,35),' '),35,' ')
   ,RPAD(NVL(SUBSTR(ven.mailing_address2,1,35),' '),35,' ')
   ,RPAD(NVL(sgeo.sc_description,' '),35,' ')
   ,RPAD(NVL(sgeo.ss_state_id,' '),2,' ')
   ,RPAD(NVL(sgeo.sz_zip5||ven.mailing_zip4,0),9,0)
   ,'    '
   ,LPAD(TO_NUMBER(ven.pee_peer_group_id),2,0)
   ,DECODE(vauth.termination_date,NULL,
     DECODE(vd.start_date,NULL,'00000000',
        DECODE(SIGN(vd.end_date - SYSDATE),-1,'00000000',
        RPAD(TO_CHAR(NVL(vd.start_date,SYSDATE),'mmddyyyy'),8,'0'))),
     DECODE(SIGN(vauth.termination_date - SYSDATE),+1,'00000000',
     RPAD(TO_CHAR(NVL(vauth.termination_date,SYSDATE),'mmddyyyy'),8,'0')))
```

```
,RPAD(NVL(SUBSTR(vb.bank_name,1,30),' '),30,' ')
,LPAD(vb.routing_number,9,0)
,RPAD(NVL(va.account_number,' '),10,' ')
```

1.14.4 EC1_PGA_FSMC.SQL

Overview

This script creates a text file of vendor peer group averages data for bank processing. All records are sent to the bank at the end of month.

1.14.4.1 Inputs

V_PEER_GROUP_AVGS

1.14.4.2 Selection

All peer group averages information are included in the output file.

1.14.4.3 Processing

Creates a text file of vendor peer group averages data for bank processing. This includes the peer group ID, FI type, and maximum redemption amount.

The detail information includes the following:

```
lpad(to_number(pga.pee_peer_group_id),2,0)
 ,lpad(nvl(substr(pga.ifit_fi_type_code,1,6),0),6,0)
 ,rpad(nvl(substr(pga.ifit_fi_type_code,7,2),'XX'),2,'XX')
 ,'**'
 '000000'
,substr(lpad(ltrim(to_char(nvl(pga.max_redemption_amt,0),'9999.99')),7,'0'),1,4)
 ,substr(lpad(ltrim(to_char(nvl(pga.max_redemption_amt,0),'9999.99')),7,'0'),6,2)
```

Also sends another line to the temporary data set, containing the total count of records, preceded by the word “TOTAL”.

The summary information includes the following:

```
'TOTAL'
,lpad(ltrim(to_char(count(*))),6,'0')
```

1.14.4.4 Outputs

ANPGAVER.DAT file

Note: For file layout properties, please refer to Appendix B

1.15 EC2.BAT

Overview

This batch file loads data into Central from each Agency.

Processing

This script first reads in the EC2.INI file to initialize and set temporary environmental variables.

Runs a truncate section to prepare tables for the Agency loading process which calls the SQL scripts:

EC_TRGOFF.SQL
EC2_SP_TRNC.SQL

For each Agency:

The script first checks to see if the Agency DMP file exists by looking for the completion flag %.don (wildcard represents the Agency) returns an 'OK' in the %.flg file and deletes the completion flag. If the .dmp file does not exist processing proceeds to the next Agency.

The following SQL scripts are then initiated:

EC2_TRNC.SQL
EC2_AGENCY.SQL
EC2B.SQL

If the dump file is missing, then it inserts a record in the e_no_eod table for next end of day run reference by funning EC2_NO_EOD.SQL script.

After the Agency dump has been completed the script runs EC2.SQL which runs the following SQL scripts:

EC2_SYNC_CLINIC_ASSIGNMENT.SQL
EC2_DUPLICATE_FI_RANGE.SQL
EC2_POPULATE_CLINIC.SQL
EC2_FOOD_UP.SQL
EC2_FI_UPD.SQL
EC2_FAILED_FI.SQL
EC2_OUTBOUND_ISSUE_FSMC.SQL
EC2_ISSUE_FSMC.SQL

Note: The Arizona WIC System's End of Day Processing doesn't generate the folloiwng reports.

Produces Vendor reports that run quartery, listed by DFDD section number. The quarterly reports are:

Section 7 - 3.1.9: Vendor Quarterly Profiles
Section 7 - 3.4.4: High Risk Vendor (AZW350, AZW355)
Section 7 - 3.5.9: Vendor Formula Redemption (AZW388)

Produces Vendor reports that run monthly, listed by DFDD section number. The monthly reports are:

Section 7 - 3.1.5: Compliance Cases and Sanctions Summary
Section 7 - 3.1.6: Monitoring and Sanctions History
Section 7 - 3.4.11: Vendor Alpha within Agency (AZW168)
Section 7 - 3.4.12: Vendor Application Data
Section 7 - 3.4.13: Vendor Authorization
Section 7 - 3.4.14: Vendor ID within State (AZW165)
Section 7 - 3.5.1: FI Type File Listing (AZW981)
Section 7 - 3.5.2: FI Type Price Update (AZW305)
Section 7 - 3.5.3: FI's Paid/Voided Index (AZW370)
Section 7 - 3.5.4: FI's Paid/Voided by Type Index (AZW370x)
Section 7 - 3.5.5: Flagged FI's (AZW320)
Section 7 - 3.5.6: Redemption Error (AZW702)
Section 7 - 3.5.7: Vendor Food Cost (AZW310)
Section 7 - 3.5.8: Vendor Food Cost - Issuing Agency Analysis (AZW315)
Section 7 - 3.5.10: Vendor Reject (AZW340)
Section 7 - 3.5.11: Vendor Report Card (AZW330)

1.16 Consolidate participant information

1.16.1 EC_TRGOFF.SQL

Overview

Sets database triggers off, so that table contents can be modified without restrictions.

1.16.1.1 Input

N/A

1.16.1.2 Selection

N/A

1.16.1.3 Processing

Alter the following tables with 'DISABLE ALL TRIGGERS'

F_ANNUAL_FACTORS
I_FOOD_INSTRUMENT_TYPES
C_FOOD_PACKAGE_PRESCRIPTIONS
C_CERTIFICATIONS
C_CLIENT_SERVICES

Alter the following tables with 'DISABLE CONSTRAINT'
I_FOOD_INSTRMENT_TYPES with constraint IFIT_IFIT_FK
I_FOOD_PACKAGE_FI_TYPES with constraint IFPFT_IFPFT_FK

1.16.1.4 Outputs

N/A

1.16.2 EC2_SP_TRNC.SQL

Overview

This script initializes the following tables by clearing all data in each of the temporary tables:

1.16.2.1 Inputs

E_I_FOOD_INSTRUMENT_TYPES
E_I_FOOD_INSTRUMENT_FOODS
E_I_FOOD_INSTRUMENTS
E_I_FI_FORMULAS
E_I_FOOD_PACKAGE_FI_TYPES
A_C_STATE_CLIENTS
E_C_STATE_CLIENTS

1.16.2.2 Selection

N/A

1.16.2.3 Processing

The script deletes all data from the tables listed above using the following commands:

```
TRUNCATE TABLE E_I_FOOD_INSTRUMENT_TYPES
TRUNCATE TABLE E_I_FOOD_INSTRUMENT_FOODS
TRUNCATE TABLE E_I_FOOD_INSTRUMENTS
TRUNCATE TABLE E_I_FI_FORMULAS
TRUNCATE TABLE E_I_FOOD_PACKAGE_FI_TYPES
TRUNCATE TABLE A_C_STATE_CLIENTS
TRUNCATE TABLE E_C_STATE_CLIENTS
```

1.16.2.4 Outputs

EC2_SP_TRNC.LOG

1.16.3 EC2_TRNC.SQL

Overview

This script initializes the following tables:

* End of day controls
* Staff members
* Organizational units
* Organizational programs
* Organizational unit phones
* Outreach organizations
* Outreach organizations LAs
* Outreach comms
* Outreach organization phones
* Outreach programs
* Staff phones
* Job descriptions
* Family economic units
* Family phones
* Family referrals
* FEU communications
* Smoking Histories
* Previous families
* Clients
* Client communications
* Client goals
* Client notes
* Client programs
* Client referrals
* Client Services
* Client NE topics
* Client services NE materials
* More Ethnic Groups
* Previous Names
* Breastfeeding Promo Issuances
* Allergy Foods
* PS Responses
* Client BP Issuances
* Certification Nutr Ed Topics
* Food Box Distributions
* Certificated peer counsels
* Contacts
* Peer Fbfends
* Counsels notes
* Incomes
* Income histories
* Economic unit members
* Immunizations

* Infant data
* Infant child medicals
* Resolutions
* Resolved clients
* Transfers information
* Transfer histories
* Certifications
* Bloodwork data
* Certification term reasons
* Dietary assessments
* Diet nutrients
* Health risk factors
* Woman medicals
* C W healths
* C W HH reasons bfends
* C I C healths
* C I C HH reasons bfends
* Case assignments
* F Wait lists
* F Wait lists contacts
* I Food packages
* I Food packages food
* I Food package prescriptions
* I Package distributions
* I BF promotional vouchers
* I BF voucher items
* I FI inventories
* I FI ranges
* I inventories
* I inventory organizational units
* V complaints
* V monthly activities
* V activity findings
* Delete statements

1.16.3.1 Inputs

A_STATE_CLIENTS
A_C_STATE_CLIENTS
A_F_POVERTY_BASES
A_EOD_CONTROLS
A_A_ACTIVITIES
A_A_APPT_ITEMS
A_AAS_CLASS_CATEGORIES
A_AAS_ITEMS
A_A_ATTEND_STATUSES
A_A_OFFICE_CLOSED
A_A_SERVICES
A_I_AGE_RANGES
A_C_ANSWERS
A_C_ANSWER_TYPES

A_C_BLOODWORK_TYPES
A_C_BREAST_PUMP_REASONS
A_C_BREAST_PUMP_TYPES
A_C_CATEGORIES
A_C_CAT_BLOOD_FACTORS
A_C_CAT_GOALS
A_C_CAT_NUTR_EDS
A_C_CAT_REFERRALS
A_C_COMMUNICATION_TYPES
A_C_CP_MESSAGES
A_C_DIAGNOSES
A_C_DIETARY_REQUIREMENTS
A_C_DIET_NUTRIENT_TYPES
A_C_DISABILITIES
A_C_EDUCATION_LEVELS
A_C_ELEVATIONS
A_C_ETHNIC_GROUPS
A_C_GOALS
A_C_HC_PAYEES
A_C_HH_QUESTIONS
A_C_HH_RESPONSES
A_C_INCOME_INTERVALS
A_C_INCOME_LEVELS
A_C_INCOME_SOURCES
A_C_INCOME_VERIFICATIONS
A_C_INFANT_STATUSES
A_C_LANGUAGES
A_C_MARITAL_STATUSES
A_C_NCHS_CLASSIFICATIONS
A_C_NCHS_TYPES
A_C_NCHS_DATA
A_C_NO_CONTACT_REASONS
A_C_NUTR_ED_MATERIALS
A_C_NUTR_ED_TOPICS
A_C_PICKUP_INTERVALS
A_C_PILOT_QUESTIONS
A_C_PILOT_STUDIES
A_C_PRIORITIES
A_C_PROOF_ADDRESSES
A_C_PROOF_IDENTITIES
A_C_PS_QUESTIONS
A_C_RACES
A_C_REASONS_BF_ENDED
A_C_RISK_FACTORS
A_C_RISK_FACTOR_DIAGNOSES
A_C_RISK_FACTOR_TYPES
A_C_RF_GOALS
A_C_RF_NUTR_EDS
A_C_RF_REFERRALS
A_C_RF_SERVICES
A_C_HH_RESPONSE_RFS
A_C_BMI_DATA
A_C_BMI_ANTHROPOMETRIC
A_C_BMI_WEIGHT_GAIN
A_C_SCHEDULE_DAYS
A_C_SMOKING_CHANGES

A_C_SMOKING_STAGES
A_C_SOURCES_HEALTH_CARE
A_C_SYMPTOMS
A_C_TERM_REASONS
A_C_TOPICS
A_C_VOTER_REGISTRATIONS
A_C_CAT_BLOODWORKS
A_C_DESIREABLE_WEIGHTS
A_C_IMMUNIZATIONS_NOT_ASSESSED
A_F_CONTROLS
A_F_CASELOAD_TYPES
A_F_FUND_SOURCES
A_F_POVERTY_LEVELS
A_F_WAIT_LIST_RESPONSES
A_I_BANK_DISPOSITIONS
A_I_CATEGORY_GROUPS
A_I_PACKAGES
A_I_PRODUCTS
A_I_REJECT_REASONS
A_I_CONTAINERS
A_I_DISPOSITIONS
A_I_UNITS_OF_MEASURE
A_I_VOID_REASONS
A_I_FOOD_GROUPS
A_I_FOODS
A_I_FOOD_DISTRIBUTIONS
A_I_MAXIMUM_FOODS
A_I_FOOD_PACKAGES
A_I_CATEGORY_GROUP_PKGS
A_I_FOOD_PACKAGE_FI_TYPES
A_I_FOOD_PACKAGE_FOODS
A_I_PACKAGE_DISTRIBUTIONS
A_I_WS_FOOD_GROUPS
A_I_FOOD_INSTRUMENT_TYPES
A_I_FOOD_INSTRUMENT_FOODS
A_C_FOOD_PKG_RISK_FACTORS
A_O_OUTREACH_ORG_TYPES
A_O_OUTREACH_COMM_TYPES
A_O_OUTREACH_ORGANIZATIONS
A_O_OUTREACH_ORG_PHONES
A_O_OUTREACH_COMMS
A_O_PROGRAMS
A_O_PROGRAM_DATES
A_O_TITLE_CATEGORIES
A_O_STAFF_TITLES
A_S_CITIES
A_S_CONTACT_METHODS
A_S_CONTACT_TITLES
A_S_COUNTIES
A_S_PHONE_TYPES
A_S_STATES
A_S_ZIPS
A_V_ACTIVITY_TYPES
A_V_APPLICATION_MILESTONES
A_V_APPLICATION_TYPES
A_V_BANK_BRANCHES

A_V_CASE_STATUSES
A_V_COLLECTION_TYPES
A_V_COMMUNICATION_TYPES
A_V_COMPLAINT_SOURCE_TYPES
A_V_COMPLAINT_STATUSES
A_V_COMPLAINT_SUBJECTS
A_V_COMPLIANCE_CASE_DESIGNATNS
A_V_COMPLIANCE_CASE_TYPES
A_V_DELIVERY_TYPES
A_V_DENIAL_REASONS
A_V_DISQUAL_REASONS
A_V_FINDING_CODES
A_V_FSP_REGIONAL_OFFICES
A_V_FSP_VIOLATION_CODES
A_V_LEGAL_FIRMS
A_V_LEGAL_REPRESENTATIVES
A_V_MILESTONE_TYPES
A_V_OWNER_TYPES
A_V_PEER_GROUPS
A_V_RISK_LEVELS
A_V_SANCTION_TYPES
A_V_STATUSES
A_V_SUSPENSION_REASONS
A_V_VIOLATION_ACTION
A_V_WHOLESALERS
A_V_WIC_CODES
A_V_OWNERS
A_V_VENDORS
A_O_STAFF_MEMBERS
A_O_ORGANIZATIONAL_UNITS
A_O_ANNUAL_WIC_COST_SUMMARIES
A_F_ANNUAL_FACTORS
A_F_BUDGETS
A_F_CASE_ASSIGNMENTS
A_F_CASELOADS
A_F_CASELOAD_RESTRICTIONS
A_F_MFR_CONTACTS
A_F_MANUFACTURERS
A_I_FI_INVENTORIES
A_I_STOCK_INVENTORIES
A_I_FOOD_INSTRUMENTS
A_I_FI_REJECT_REASONS
A_C_RESOLUTIONS
A_C_RESOLVED_CLIENTS
A_S_GEO_LOCATIONS;
A_S_PRINTER_ADDRESSES;
A_DEL_STATEMENTS
R_S_CLIENT_ARCHIVES
PROMPT table containing clients who have been retrieved from the archives
R_ARCHIVED_CLIENTS
PROMPT table containing clients who have been archived at central

1.16.3.2 Selection

N/A

1.16.3.3 Processing

The script deletes all data from the tables listed above using the following command for each table:

TRUNCATE TABLE TABLE_NAME

1.16.3.4 Outputs

N/A

1.16.4 EC2_AGENCY.SQL

Overview

Executes SQL scripts for EC2.BAT.

1.16.4.1 Input

env_variables

1.16.4.2 Selection

env_variables
where code = ‘EOD_BY’

1.16.4.3 Processing

update eod_controls, setting TO_DATE to the system date
update env_variables, setting env_value to ‘EC2' where code = ‘EOD_BY’

run SQL script EC2_UPD_E_C_CLIENTS
run SQL script EC2_DELTRIG
run SQL script EC2_IN_TAB
run SQL script EC2_TAB_UP
run SQL script EC2_DELTRIG_AU
run SQL script EC2_ST_CLIENTS_SUM

1.16.4.4 Output

env_varibles

Log file: EC2_AGENCY.LOG

EC2_UPD_E_C_CLIENTS.SQL

Overview

This script updates the last_first_bdate column of the table e_c_clients table to speed up the dual enrollment process.

1.16.4.5 Inputs

E_C_CLIENTS

1.16.4.6 Selections

N/A

1.16.4.7 Processing

The script updates the last_first_bdate colun of the e_c_clients table using the following command.

```
UPDATE E_C_CLIENTS
SET LAST_FIRST_BDATE=SUBSTR (LAST_NAME, 1, 4) ||
                        SUBSTR(first_name, 1, 6) ||
                        to_char(birth_date, 'MMYYYY')
                where compliance_flag = 'N'
```

1.16.4.8 Outputs

N/A

1.16.5 EC2_DELTRIG.SQL

Overview

This script deletes data from the central database that had been deleted from the local agency database since the last end of day run excluding the c_family_economic_units and c_clients table.

1.16.5.1 Inputs

E_DEL_STATEMENTS
EOD_CONTROLS

1.16.5.2 Selection

E_DEL_STATEMENTS WHERE target_proc_date IS NULL
AND table_name not in('C_FAMILY_ECONOMIC_UNITS', 'C_CLIENTS')
ORDER BY origin_agcy, del_seq

1.16.5.3 Processing

Loop through each E_DEL_STATEMENTS record meeting the selection criteria

Execute the SQL delete statement found in the del_statement column.

End loop

UPDATE e_del_statements SET target_proc_date = eod_controls.to_date
WHERE target_proc_date IS NULL
AND table_name not in('C_FAMILY_ECONOMIC_UNITS', 'C_CLIENTS')

1.16.5.4 Outputs

Log file (EC2_AGENCY.LOG) containing a list of all delete statements executed.

1.16.6 EC2_DELTRIG_AU.SQL

Overview

This script deletes data from the central database that had been deleted from the local agency database since the last end of day run for the c_family_economic_units and c_clients tables only.

1.16.6.1 Inputs

E_DEL_STATEMENTS
EOD_CONTROLS

1.16.6.2 Selection

E_DEL_STATEMENTS WHERE target_proc_date is NULL
AND table_name = ‘CLIENTS’
ORDER BY origin_agcy, del_seq

E_DEL_STATEMENTS WHERE target_proc_date IS NULL
AND table_name = 'C_FAMILY_ECONOMIC_UNITS'
ORDER BY origin_agcy, del_seq

1.16.6.3 Processing

Loop through each E_DEL_STATEMENTS record meeting the selection criteria for c_clients first.

Execute the SQL delete statement found in the del_statement column.

End loop

Loop through each E_DEL_STATEMENTS record meeting the selection criteria.

Execute the SQL delete statement found in the del_statement column.

End loop

UPDATE e_del_statements SET target_proc_date = eod_controls.to_date
WHERE target_proc_date IS NULL
AND table_name in ('C_FAMILY_ECONOMIC_UNITS', ‘C_CLIENTS’)

1.16.6.4 Outputs

Log file (EC2_AGENCY.LOG) containing a list of all delete statements executed.

1.16.7 EC2_IN_TAB.SQL

Overview

This script inserts client information stored at the central database that has been created at the local agencies, copied to the central database, and stored in the E_* tables. It compares the created date in each of the following tables to the last day that the end of day processes ran successfully.

- Staff members
- Staff time studies
- Organizational units
- Organizational programs
- Organizational unit phones
- Outreach organizations
- Outreach_org las
- Outreach comms
- Outreach organization phones
- Outreach programs
- Staff phones
- Job descriptions
- Family economic units
- Family phones
- Family referrals
- FEU communications
- C Food Package Prescriptions
- Client PS Responses
- C_food_box distributions
- Clients
- Client BP Issuances
- Client More Ethnic Groups
- Client Previous Names
- Client communications
- Client goals
- Client notes
- Client programs
- Client referrals
- Client Services
- Client NE topics
- Client services NE materials
- Cert Nutr Eds
- Certificated peer counsels
- Contacts
- C Allergy foods
- Peer Fbfends
- Counsels notes

- Incomes
- Income histories
- Economic unit members
- Immunizations
- Infant data
- Infant child medicals
- Resolutions
- Resolved clients
- Transfers information
- Transfer histories
- Certifications
- Bloodwork data
- Certification term reasons
- Dietary assessments
- Diet nutrients
- Health risk factors
- Woman medicals
- C B W healths
- C B W HH reasons bfends
- C I C healths
- C I C HH reasons bfends
- C P healths
- C infant child medicals
- C woman medicals
- Case assignments
- F Wait lists
- F Wait lists contacts
- I Food packages
- I Food packages food
- I Food package prescriptions
- I Package distributions
- I BF promotional vouchers
- I BF voucher items
- I FI inventories
- I FI ranges
- I FI formulas
- I inventories
- I inventory organizational units
- I stock inventories
- Batch Runs
- Printer Addresses
- V complaints
- V monthly activities
- V activity findings
- Delete statements

1.16.7.1 Inputs

E_EOD_CONTROLS
E_O_STAFF_MEMBERS
E_O_ORGANIZATIONAL_UNITS
E_O_ORG_PROGRAMS
E_O_ORG_UNIT_PHONES
E_O_OUTREACH_ORGANIZATIONS
E_O_OUT_ORG_LAS
E_O_OUTREACH_COMMS
E_O_OUTREACH_ORG_PHONES
E_O_OUTREACH_PROGRAMS
E_O_STAFF_PHONES
E_O_JOB_DESCRIPTIONS
E_C_FAMILY_ECONOMIC_UNITS
E_C_FAMILY_PHONES
E_C_FAM_REFERRALS
E_C_FEU_COMMUNICATIONS
E_C_SMOKING_HISTORIES
E_C_PREV_FAMILIES
E_C_CLIENTS
E_C_CERT_PEER_COUNSELS
E_C_CONTACTS
E_C_PEER_RBFENDS
E_C_P_COUNS_NOTES
E_C_CLIENT_COMMUNICATIONS
E_C_CLIENT_GOALS
E_C_CLIENT_NOTES
E_C_CLIENT_PROGS
E_C_CLIENT_REFERRALS
E_C_CLIENT_SERVICES
E_C_CLIENT_NE_TOPICS
E_C_CLIENT_SVC_NE_MATERIALS
E_C_MORE_ETHNIC_GROUPS
E_C_PREV_NAMES
E_C_BF_PROMO_ISSUANCES
E_C_ALLERGY_FOODS
E_C_W_HEALTHS
E_C_W_HH_REASONS_BFENDS
E_C_PS_RESPONSES
E_C_CLIENT_BP_ISSUANCES
E_C_INCOME_HISTORIES
E_C_ECONOMIC_UNIT_MEMBERS
E_C_INCOMES
E_C_INFANT_DATA
E_C_RESOLUTIONS
E_C_RESOLVED_CLIENTS
E_C_TRANSFERS_INFO
E_C_TRANSFER_HISTORIES
E_C_CERTIFICATIONS
E_C_CERT_NUTR_EDS
E_C_FOOD_BOX_DISTS

E_C_BLOODWORK_DATA
E_C_CERT_TERM_REASONS
E_C_DIETARY_ASSESSMENTS
E_C_DIET_NUTRIENTS
E_C_HEALTH_RISK_FACTORS
E_C_INFANT_CHILD_MEDICALS
E_C_WOMAN_MEDICALS
E_C_I_C_HEALTHS
E_C_I_C_HH_REASONS_BFENDS
E_F_CASE_ASSIGNMENTS
E_F_WAIT_LISTS
E_F_WAIT_LIST_CONTACTS
E_I_FOOD_PACKAGES
E_I_FOOD_PACKAGE_FOODS
E_I_PACKAGE_DISTRIBUTIONS
E_C_FOOD_PACKAGE_PRESCRIPTIONS
E_I_BF_PROMO_VCHRS
E_I_BF_VCHR_ITEMS
E_I_FI_INVENTORIES
E_I_INVENTORIES
E_I_INV_ORG_UNITS
E_V_COMPLAINTS
E_V_MON_ACTIVITIES
E_V_ACTIVITY_FINDINGS
E_DEL_STATEMENTS

1.16.7.2 Selection

Only records that have been created since the last successful end of day process (This selection is done for each of the above tables):

TABLE_NAME WHERE date_created > (SELECT good_proc_date FROM EODADM.e_eod_controls)

1.16.7.3 Processing

For each table listed in the outputs section the system inserts the records found in the corresponding E_* table found in the inputs section
Loop

> Insert into TABLE_NAME (select * from E_TABLE_NAME where date_created > (SELECT good_proc_date FROM EODADM.e_eod_controls);

End Loop

For example the insert statement for the O_STAFF_MEMBERS table is:

Insert into O_STAFF_MEMBERS as select * from E_O_STAFF_MEMBERS;
WHERE date_created > (select good_proc_date from eodadm.e_eod_controls);

1.16.7.4 Outputs

O_STAFF_MEMBERS
O_STAFF_TIME_STUDIES
O_ORG_PROGRAMS
O_ORG_UNIT_PHONES
O_OUTREACH_ORGANIZATIONS
O_OUT_ORG_LAS
O_OUTREACH_COMMS
O_OUTREACH_ORG_PHONES
O_OUTREACH_PROGRAMS
O_STAFF_PHONES
O_JOB_DESCRIPTIONS
C_FAMILY_ECONOMIC_UNITS
C_FAMILY_PHONES
C_FAM_REFERRALS
C_FEU_COMMUNICATIONS
C_SMOKING_HISTORIES
C_CLIENTS
C_CLIENTS
C_PREV_FAMILIES
C_CERT_PEER_COUNSELS
C_CONTACTS
C_PEER_RBFENDS
C_P_COUNS_NOTES
C_CLIENT_COMMUNICATIONS
C_CLIENT_GOALS
C_CLIENT_NOTES
C_CLIENT_PROGS
C_CLIENT_REFERRALS
C_CLIENT_SERVICES
C_CLIENT_NE_TOPICS
C_CLIENT_SVC_NE_MATERIALS
C_MORE_ETHNIC_GROUPS
C_PREV_NAMES
C_BF_PROMO_ISSUANCES
C_ALLERGY_FOODS
C_W_HEALTHS
C_W_HH_REASONS_BFENDS
C_PS_RESPONSES
C_CLIENT_BP_ISSUANCES
C_INCOME_HISTORIES
C_ECONOMIC_UNIT_MEMBERS
C_INCOMES
C_INFANT_DATA
C_RESOLVED_CLIENTS
C_RESOLUTIONS
C_TRANSFERS_INFO

C_TRANSFER_HISTORIES
C_CERTIFICATIONS
C_CERT_NUTR_EDS
C_FOOD_BOX_DISTS
C_BLOODWORK_DATA
C_CERT_TERM_REASONS
C_DIETARY_ASSESSMENTS
C_DIET_NUTRIENTS
C_HEALTH_RISK_FACTORS
C_INFANT_CHILD_MEDICALS
C_WOMAN_MEDICALS
C_I_C_HEALTHS
C_I_C_HH_REASONS_BFENDS
F_CASE_ASSIGNMENTS
F_WAIT_LISTS
F_WAIT_LIST_CONTACTS
I_FOOD_PACKAGES
I_FOOD_PACKAGE_FOODS
I_PACKAGE_DISTRIBUTIONS
C_FOOD_PACKAGE_PRESCRIPTIONS
I_BF_PROMO_VCHRS
I_BF_VCHR_ITEMS
I_FI_INVENTORIES
I_INVENTORIES
I_STOCK_INVENTORIES
I_INV_ORG_UNITS
V_COMPLAINTS
V_MON_ACTIVITIES
V_ACTIVITY_FINDINGS
F_CASELOAD_RESTRICTIONS
I_BATCH_RUNS
S_PRINTER_ADDRESSES
DEL_STATEMENTS

1.16.8 EC2_TAB_UP.SQL

Overview

This script updates client information stored at the central database that has been modified at the local agencies, copied to the central database, and stored in the E_* tables. It compares the modified date in each of the following tables to the last day that the end of day processes ran successfully.

- Staff members
- Staff time studies
- Organizational units
- Organizational programs
- Organizational unit phones
- Outreach organizations
- Outreach_org las
- Outreach comms
- Outreach organization phones
- Outreach programs
- Staff phones
- Job descriptions
- Family economic units
- Family phones
- Family referrals
- FEU communications
- C Food Package Prescriptions
- Client PS Responses
- C_food_box distributions
- Clients
- Client BP Issuances
- Client More Ethnic Groups
- Client Previous Names
- Client communications
- Client goals
- Client notes
- Client programs
- Client referrals
- Client Services
- Client NE topics
- Client services NE materials
- Cert Nutr Eds
- Certificated peer counsels
- Contacts
- C Allergy foods
- Peer Fbfends
- Counsels notes

- Incomes
- Income histories
- Economic unit members
- Immunizations
- Infant data
- Infant child medicals
- Resolutions
- Resolved clients
- Transfers information
- Transfer histories
- Certifications
- Bloodwork data
- Certification term reasons
- Dietary assessments
- Diet nutrients
- Health risk factors
- Woman medicals
- C B W healths
- C B W HH reasons bfends
- C I C healths
- C I C HH reasons bfends
- C P healths
- C infant child medicals
- C woman medicals
- Case assignments
- F Wait lists
- F Wait lists contacts
- I Food packages
- I Food packages food
- I Food package prescriptions
- I Package distributions
- I BF promotional vouchers
- I BF voucher items
- I FI inventories
- I FI ranges
- I FI formulas
- I inventories
- I inventory organizational units
- I stock inventories
- Batch Runs
- Printer Addresses
- V complaints
- V monthly activities
- V activity findings
- Delete statements

1.16.8.1 Inputs

E_EOD_CONTROLS
E_O_STAFF_MEMBERS
E_O_ORGANIZATIONAL_UNITS
E_O_ORG_PROGRAMS
E_O_ORG_UNIT_PHONES
E_O_OUTREACH_ORGANIZATIONS
E_O_OUT_ORG_LAS
E_O_OUTREACH_COMMS
E_O_OUTREACH_ORG_PHONES
E_O_OUTREACH_PROGRAMS
E_O_STAFF_PHONES
E_O_JOB_DESCRIPTIONS
E_C_FAMILY_ECONOMIC_UNITS
E_C_FAMILY_PHONES
E_C_FAM_REFERRALS
E_C_FEU_COMMUNICATIONS
E_C_SMOKING_HISTORIES
E_C_PREV_FAMILIES
E_C_CLIENTS
E_C_CERT_PEER_COUNSELS
E_C_CONTACTS
E_C_PEER_RBFENDS
E_C_P_COUNS_NOTES
E_C_CLIENT_COMMUNICATIONS
E_C_CLIENT_GOALS
E_C_CLIENT_NOTES
E_C_CLIENT_PROGS
E_C_CLIENT_REFERRALS
E_C_CLIENT_SERVICES
E_C_CLIENT_NE_TOPICS
E_C_CLIENT_SVC_NE_MATERIALS
E_C_MORE_ETHNIC_GROUPS
E_C_PREV_NAMES
E_C_BF_PROMO_ISSUANCES
E_C_ALLERGY_FOODS
E_C_W_HEALTHS
E_C_W_HH_REASONS_BFENDS
E_C_PS_RESPONSES
E_C_CLIENT_BP_ISSUANCES
E_C_INCOME_HISTORIES
E_C_ECONOMIC_UNIT_MEMBERS
E_C_INCOMES
E_C_INFANT_DATA
E_C_RESOLUTIONS
E_C_RESOLVED_CLIENTS
E_C_TRANSFERS_INFO
E_C_TRANSFER_HISTORIES
E_C_CERTIFICATIONS
E_C_CERT_NUTR_EDS
E_C_FOOD_BOX_DISTS

E_C_BLOODWORK_DATA
E_C_CERT_TERM_REASONS
E_C_DIETARY_ASSESSMENTS
E_C_DIET_NUTRIENTS
E_C_HEALTH_RISK_FACTORS
E_C_INFANT_CHILD_MEDICALS
E_C_WOMAN_MEDICALS
E_C_I_C_HEALTHS
E_C_I_C_HH_REASONS_BFENDS
E_F_CASE_ASSIGNMENTS
E_F_WAIT_LISTS
E_F_WAIT_LIST_CONTACTS
E_I_FOOD_PACKAGES
E_I_FOOD_PACKAGE_FOODS
E_I_PACKAGE_DISTRIBUTIONS
E_C_FOOD_PACKAGE_PRESCRIPTIONS
E_I_BF_PROMO_VCHRS
E_I_BF_VCHR_ITEMS
E_I_FI_INVENTORIES
E_I_INVENTORIES
E_I_INV_ORG_UNITS
E_V_COMPLAINTS
E_V_MON_ACTIVITIES
E_V_ACTIVITY_FINDINGS
E_DEL_STATEMENTS

1.16.8.2 Selection

Only records that have been updated since the last successful end of day process (This selection is done for each of the above tables):

TABLE_NAME WHERE date_modified > (SELECT good_proc_date FROM EODADM.e_eod_controls)

1.16.8.3 Processing

The system updates each table listed in the outputs section to match the records found in the corresponding E_* table from the inputs section.

Loop

> Update TABLE_NAME set (column list) = (select * from E_TABLE_NAME where date_modified > (SELECT good_proc_date FROM EODADM.e_eod_controls);

End Loop

1.16.8.4 Outputs

O_STAFF_MEMBERS
O_STAFF_TIME_STUDIES
O_ORG_PROGRAMS
O_ORG_UNIT_PHONES
O_OUTREACH_ORGANIZATIONS
O_OUT_ORG_LAS
O_OUTREACH_COMMS
O_OUTREACH_ORG_PHONES
O_OUTREACH_PROGRAMS
O_STAFF_PHONES
O_JOB_DESCRIPTIONS
C_FAMILY_ECONOMIC_UNITS
C_FAMILY_PHONES
C_FAM_REFERRALS
C_FEU_COMMUNICATIONS
C_SMOKING_HISTORIES
C_CLIENTS
C_CLIENTS
C_PREV_FAMILIES
C_CERT_PEER_COUNSELS
C_CONTACTS
C_PEER_RBFENDS
C_P_COUNS_NOTES
C_CLIENT_COMMUNICATIONS
C_CLIENT_GOALS
C_CLIENT_NOTES
C_CLIENT_PROGS
C_CLIENT_REFERRALS
C_CLIENT_SERVICES
C_CLIENT_NE_TOPICS
C_CLIENT_SVC_NE_MATERIALS
C_MORE_ETHNIC_GROUPS
C_PREV_NAMES
C_BF_PROMO_ISSUANCES
C_ALLERGY_FOODS
C_W_HEALTHS
C_W_HH_REASONS_BFENDS
C_PS_RESPONSES
C_CLIENT_BP_ISSUANCES
C_INCOME_HISTORIES
C_ECONOMIC_UNIT_MEMBERS
C_INCOMES
C_INFANT_DATA
C_RESOLVED_CLIENTS
C_RESOLUTIONS
C_TRANSFERS_INFO
C_TRANSFER_HISTORIES
C_CERTIFICATIONS

C_CERT_NUTR_EDS
C_FOOD_BOX_DISTS
C_BLOODWORK_DATA
C_CERT_TERM_REASONS
C_DIETARY_ASSESSMENTS
C_DIET_NUTRIENTS
C_HEALTH_RISK_FACTORS
C_INFANT_CHILD_MEDICALS
C_WOMAN_MEDICALS
C_I_C_HEALTHS
C_I_C_HH_REASONS_BFENDS
F_CASE_ASSIGNMENTS
F_WAIT_LISTS
F_WAIT_LIST_CONTACTS
I_FOOD_PACKAGES
I_FOOD_PACKAGE_FOODS
I_PACKAGE_DISTRIBUTIONS
C_FOOD_PACKAGE_PRESCRIPTIONS
I_BF_PROMO_VCHRS
I_BF_VCHR_ITEMS
I_FI_INVENTORIES
I_INVENTORIES
I_STOCK_INVENTORIES
I_INV_ORG_UNITS
V_COMPLAINTS
V_MON_ACTIVITIES
V_ACTIVITY_FINDINGS
F_CASELOAD_RESTRICTIONS
I_BATCH_RUNS
S_PRINTER_ADDRESSES
DEL_STATEMENTS

1.16.9 EC2_ST_CLIENTS_SUM.SQL

Overview

The c_state_clients table is updated based upon records in the family, client, and certification temporary tables that came up from the agencies. The c_state_clients table is sent down during the ec4 process for dual enrollment checks at the agencies.

1.16.9.1 1.17.9.1 Inputs

e_c_family_economic_units
c_family_economic_units
e_c_clients
c_clients
e_c_certifications
c_certifications

1.16.9.2 Selection

All records in e_c_family_economic_units, e_c_clients, and e_c_certifications are selected.

1.16.9.3 Processing

Insert client records into c_state_clients that exist in e_c_clients but not c_state_clients.
Update all records found in c_state_clients with a corresponding record in e_c_family_economic_units, e_c_clients, or e_c_certifications.

1.16.9.4 Outputs

c_state_clients

1.16.10 EC2B.SQL

Overview

Executes SQL scripts for EC2.BAT.

1.16.10.1 Inputs

env_variables

1.16.10.2 Selection

env_variables
where code = ‘EOD_BY’

1.16.10.3 Processing

update eod_controls, setting TO_DATE to the system date
update env_variables, setting env_value to ‘EC2b' where code = ‘EOD_BY’

run SQL script EC2_CL_REC_PROCESS
run SQL script EC2_DEL_CL_FUT_RECS
run SQL script EC2_DUAL_ENROLL

1.16.10.4 Output

env_varibles

Log file: EC2b_SQL.LOG

1.16.11 EC2_DEL_CL_FUT_RECS.SQL

Overview

This script is used to delete future records of a client with respect to the old Local Agency when the client is transferred, or when the client is terminated. This needs to be executed at central site after EOD data from all local agencies are received and before EC2_STATE_CTI.SQL is run.

1.16.11.1 Inputs

C_CLIENT_SERVICES
C_CLIENT_SVC_NE_MATERIALS
C_CLIENT_NE_TOPICS

1.16.11.2 Selection

Select all clients from the C_CLIENT_SERVICES table where the following criteria are met: C_CLIENT_SERVICES.CLIENT_ID equals ID of client to be transferred, the service date is on or after the termination date, the appointment flag is set to 'Y', and the date created and date modified are before the termination date.

1.16.11.3 Processing

For all clients to be transferred:

Delete record from C_CLIENT_SVC_NE_MATERIALS table,
Delete record from C_CLIENT_NE_TOPICS table,
Delete record from C_CLIENT_SERVICES table.

1.16.11.4 Outputs

C_CLIENT_SERVICES
C_CLIENT_SVC_NE_MATERIALS
C_CLIENT_NE_TOPICS

1.16.12 EC_TRGON.SQL

Overview

Sets database triggers on, so that triggers and foreign keys are re-enabled.

1.16.12.1 Input

N/A

1.16.12.2 Selection

N/A

1.16.12.3 Processing

Alter the following tables with 'ENABLE ALL TRIGGERS'

F_ANNUAL_FACTORS
I_FOOD_INSTRUMENT_TYPES
C_FOOD_PACKAGE_PRESCRIPTIONS
C_CERTIFICATIONS
C_CLIENT_SERVICES

Alter the following tables with 'ENABLE CONSTRAINT'

I_FOOD_INSTRMENT_TYPES with constraint IFIT_IFIT_FK
I_FOOD_PACKAGE_FI_TYPES with constraint IFPFT_IFPFT_FK

1.16.12.4 Output

N/A

1.17 Financial - process totals and store in F_CASELOAD table. Totals are based on priority, language, caseload type, ethnic group and poverty level

1.17.1 EC2_CL_REC_PROCESS.SQL

Overview

This end of day script is used to insert updated records into caseload type detail and caseloads for each certification record that has been created or modified. For more information, see the Financial Management Introduction.

1.17.1.1 Inputs

env_variables
eod_controls
e_eod_controls
e_c_certifications
c_clients
c_family_economic_units
f_caseload_type_details
f_incomes_to_poverties
f_caseloads

1.17.2 PROCEDURE EODP_CL_REC_PROC_P1

This EOD script is to process any changes in c_certification / c_client data. Inserts and Updates are performed on the F_CASELOAD_TYPE_DETAILS and F_CASELOADS tables. Executed at Central site only.

1.17.2.1 Processing

Select all C_CERTIFICATIONS records where date_created or date_modified is between EOD_CONTROLS.FROM_DATE and EOD_CONTROLS.TO_DATE.

If C_CERTIFICATIONS.TERMINATION_DATE is null, then
For all months in the C_CERTIFICATION record ranging from the CERT_START_DATE to the CERT_END_DATE

Select the max(F_INCOMES_TO_POVERTIES.FPL_ID) for the Participant's income and family size.

Select the F_CASELOAD_TYPE_DETAILS record for the Federal Fiscal Month and Year, Program, Clinic, Category, Priority, Language, Ethnic Group, Migrant Flag, Refugee Flag, Adjunctive Eligibility Flag, Income, Family Size, and

Poverty Level from the C_CERTIFICATIONS record where PARTICIPANT_FLAG = 'N'.

If the F_CASELOAD_TYPE_DETAILS record does not exist, then
Insert the information from the select statement into a new record.

If EOD_CONTROLS.IPL_DATE is in the same or a later month than the C_CERTIFICATION month and the C_CERTIFICATION month is in the same month as the month when the C_CERTIFICATION record was last modified or created, then
Run the EODP_I_U_CASELOAD_P1('ENROLLEE')

If the MIGRANT_FLAG for the C_CERTIFICATIONS record equals 'Y', then
Run the EODP_I_U_CASELAOD_P1('MIGRANT')

If the REFUGEE_FLAG for the C_CERTIFICATIONS record equals 'Y', then
Run the EODP_I_U_CASELAOD_P1('REFUGEE')

If the C_CERTIFICATION record is for a Category of 'IEN' with a Rec Status = 'A' and a Mother's ID where that Mother's Client ID is in a current C_FOOD_PACKAGE_PRESCRIPTION record, then
Run the EODP_I_U_CASELAOD_P1('PARTICIPANT')

1.17.3 PROCEDURE EODP_I_U_CASELOAD_P1

This EOD function is used to insert update changes into the caseload table. It is called by the client record processing procedure (EODP_CL_REC_PROC_P1).

1.17.3.1 Processing

For the Caseload Type passed in the parameter of the procedure call
Select the Client Count from F_CASELOADS for the Fiscal Month and Year, Caseload Type, Program, Clinic, Language, Priority, Category, Ethnic Group, and Poverty Level from the F_CASELOAD_TYPE_DATAILS record in the EODP_C1_REC_PROC_P1 procedure.

If the F_CASELOAD record does not exist, then
Insert the information from the select statement into a new record. The Client Count will be set to 1.
Else
Update the F_CASELOAD record by incrementing the Client Count by 1.

1.17.3.2 Outputs

f_caseloads

f_caseload_type_details

1.17.4 EC2.SQL

Overview

Executes SQL scripts for EC2.BAT.

1.17.4.1 Input

env_variables

1.17.4.2 Selection

env_variables
where code = 'EOD_BY'

1.17.4.3 Processing

update eod_controls, setting TO_DATE to the system date
update env_variables, setting env_value to 'EC2' where code = 'EOD_BY'

run SQL script EC2_SYNC_ASSIGNMENT
run SQL script EC2_DUPLICATE_FI RANGE
run SQL script EC2_POPULATE_CLINIC
run SQL script EC2_FOOD_UP
run SQL script EC2_FI_UPD
run SQL script EC2_FAILED_FI.SQL
run SQL script EC2_OUTBOUND_ISSUE_FSMC.SQL
run SQL script EC2_ISSUE_FSMC

1.17.4.4 Output

env_varibles

Log files:
EC2_SYNC_CLINIC.LOG
EC2_FOOD_UP.LOG
EC2_FI_UPD.LOG
ISSUEANDAT

1.18 Update caseload assignment information from the clinics

1.18.1 EC2_SYNC_CLINIC_ASSIGNMENT.SQL

Overview

This script is used at the state site for end of day processes to synchronize the caseload data. The caseload assignments are re-allocated by the local agency to their clinics. After this processing the records are deleted from the eodadm.e_f_case_assignment table.

1.18.1.1 Inputs

eodadm.e_f_case_assignments
f_case_assignments

1.18.1.2 Selection

Checks to see if caseloads records exist at the local agency level that have been moved up to the state level and locks those records for update. If they do, then the caseload records are updated. If they don't, then new caseload records are inserted.

Select 'x'
From f_case_assignments
Where fbu_ffy_month equals eodadm.e_f_case_assignments.fbu_ffy_month
And fbu_fff_ffy equals eodadm.e_f_case_assignments.fbu_fff.ffy
And ou_seq_id equals eodadm.e_f_case_assignments.category_code
And program equals eodadm.e_f_case_assignments.program
FOR UPDATE OF ASSIGNED

1.18.1.3 Processing

If f_case_assignments is found, then
Update f_case_assignments and
Set assigned equal to eodadm.e_f_case_assignments.assigned
alloc_factor equal to eodadm.e_f_case_assignments.alloc_factor
date_modified_by equal to eodadm.e_f_case_assignments.date_modified
note equal to eodadm.e_f_case_assignments.note
Else
Insert INTO f_case_assignment the following values
eodadm.e_f_case_assignments.fbu_ffy_month
eodadm.e_f_case_assignments.fbu_fff_ffy
eodadm.e_f_case_assignments.category.code
eodadm.e_f_case_assignments.assigned
eodadm.e_f_case_assignments.alloc_factor

eodadm.e_f_case_assignments.created_by
eodadm.e_f_case_assignments.date_created
eodadm.e_f_case_assignments.date_modified
eodadm.e_f_case_assignments.modified_by
eodadm.e_f_case_assignments.note
eodadm.e_f_case_assignments.program

Delete from eodadm.a_f_case_assignments

1.18.1.4 Outputs

eodadm.a_f_case_assignments
f_case_assignments

1.19 Consolidate and update all food instrument data to calculate FI obligation value

1.19.1 EC2_FOOD_UP.SQL

Overview

This sql script will call the ec2_fi_upd function from the eod_fi_process database package. This function moves all the food instruments that were created at the clinic level up to the state agency and into the I_BANK_REPORTS table for bank processing. Initially it will call the procedure: FI_TYPE_PROCESS which will copy FI tyeps from the Local Agency to the State database.

1.19.1.1 Inputs

eodadm.e_i_food_instrument_types
eodadm.e_i_food_instrument_foods
i_food_instrument_types
i_food_instrument_foods

1.19.1.2 Processing

For each record in EODADM.E_I_FOOD_INSTRUMENTS
If the food instrument does not exist in AIM.I_FOOD_INSTRUMENTS, then
Insert the FI from EODADM.E_I_FOOD_INSTRUMENTS into AIM.I_FOOD_INSTRUMENTS.

Insert the FI from EODADM.E_I_FOOD_INSTRUMENTS into AIM.I_BANK_REPORTS.

Delete the FI from EODADM.E_I_FOOD_INSTRUMENTS.

Else
If the AIM and EODADM FIs are different, then
Update AIM.I_FOOD_INSTRUMENTS with the data from EODADM.E_I_FOOD_INSTRUMENTS

Insert the FI from EODADM.E_I_FOOD_INSTRUMENTS into AIM.I_BANK_REPORTS

1.19.1.3 Outputs

i_food_instruments
i_bank_reports

1.19.2 EC2_FI_UPD.SQL

Overview

This sql script calls the ec2_fi_updates procedure, which updates the financial management tables with the obligation information on the food instruments that were created at the clinic/local agency level and moved to the state agency by the ec2_food_up sql file.

1.19.2.1 Inputs

eod_controls
i_bank_reports
f_obligations
c_certifications
c_food_package_prescriptions
i_food_instrument_foods
f_reb_contracts
f_outlays
f_reb_details
o_organizational_units
i_dispositions
f_annual_factors
f_annual_category_factors
f_budgets
i_food_instruments
f_obligations
i_bank_reports

1.19.2.2 Processing

For all records in I_BANK_REPORTS where the I_BANK_REPORTS.REC_STATUS != 'P' and I_BANK_REPORTS.POST_CODE = 'Y' and
I_BANK_REPORTS.IVR_VOID_REASON_CODE != ('A','E') and
I_BANK_REPORTS.SEND_CODE != 'L'

If I_BANK_REPORTS.REC_STATUS = 'I' or if I_BANK_REPORTS.REC_STATUS = 'U' and I_BANK_REPORTS.MISSING_ISSUANCE_FLAG = 'Y', then

If I_BANK_REPORTS.ISSUE_DATE is not null and
I_BANK_REPORTS.OU_SEQ_ID is not the State Agency and
I_BANK_REPORTS.IFIT_FI_TYPE_CODE is not null, then

Update F_CASELOAD_TYPE_DETAILS.PARTICIPANT_FLAG = 'Y' for the F_CASELOAD_TYPE_DETAILS.CC_CLIENT_ID of the I_BANK_REPORTS record.

Select the Client Count from F_CASELOADS for the Fiscal Month and Year, Program, Clinic, Language, Priority, Category, Ethnic Group, and

Poverty Level where the Caseload Type = 'PARTICIPANT' from the F_CASELOAD_TYPE_DETAILS record that was updated.

If the F_CASELOAD record does not exist, then
Insert the information from the select statement into a new record. The Client Count will be set to 1.
Else
Update the F_CASELOAD record by incrementing the Client Count by 1.

Fetch the F_OBLIGATIONS record for the I_BANK_REPORTS.OU_SEQ_ID and I_BANK_REPORTS.IFIT_FI_TYPE_CODE where I_BANK_REPORTS.FIRST_DATE_TO_USE is in F_OBLIGATIONS.FBU_FFY_MONTH and F_OBLIGATIONS.FBU_FFF_FFY.
If the F_OBLIGATIONS record is not found, then
Insert into F_OBLIGATIONS
The Category and Priority are derived from I_BANK_REPORTS.CC1_CLIENT_ID.

The FI Type is set to I_BANK_REPORTS.IFIT_FI_TYPE_CODE

The Fiscal Year and Month are derived from I_BANK_REPORTS.FIRST_DATE_TO_USE

The Organizational Unit is set to I_BANK_REPORTS.OU_SEQ_ID

The FI Created Count and FI Obligated Count are set to '1'.

If three months of prior data exist in the F_OUTLAYS table for months where F_BUDGETS.CLOSED_OUT_MONTH_FLAG = 'Y' then F_OBLIGATIONS.EST_REDEEM_FACTOR is calculated from the actual redemption rate for those three months. This is defined as the (SUM(F_OUTLAYS.REDEEMED_FI_COUNT) for the three months) / (SUM(F_OBLIGATIONS.FI_CREATED_COUNT) - (SUM(F_OUTLAYS.ALL_VOIDS) - SUM(F_OUTLAYS.STALE_DATED_VOIDS))). This is the total number of FIs redeemed during the three month period divided by the total number of FIs that were created during the three month period that were either unvoided or voided for stale date purposes only. If three months worth of closed out data does not exist,

then
F_OBLIGATIONS.EST_REDEEM_FACTOR
is set to
F_ANNUAL_FACTORS.START_REDEEM_F
ACTOR.

If three months of prior data exist in the
F_OUTLAYS table for months where
F_BUDGETS.CLOSED_OUT_MONTH_FLA
G = 'Y' then
F_OBLIGATIONS.EST_PER_FI_VALUE is
calculated from the actual per FI value for those
three months. This is defined as
(SUM(F_OUTLAYS.REDEEMED_FI_VALUE
) for the three months) /
(SUM(F_OUTLAYS.REDEEMED_FI_COUNT
) for the three months). If three months worth of
closed out data does not exist, then
F_OBLIGATIONS.EST_PER_FI_VALUE is
set to
F_ANNUAL_FACTORS.START_FPKG_COS
T. F_OBLIGATIONS.ADJ_PER_FI_VALUE is
set to
F_OBLIGATIONS.EST_PER_FI_VALUE *
F_OBLIGATIONS.EST_REDEEM_FACTOR.

Else

Update F_OBLIGATIONS
Increment FI Created Count and FI Obligated Count by 1.

Update I_FOOD_INSTRUMENTS
Set the Obligation Amount to
F_OBLIGATIONS.EST_PER_FI_VALUE.

If the I_BANK_REPORTS record contains an FI Type that consists of a Food for which an F_REB_CONTRACTS.IF_FOOD_ID exists, then

Update I_FOOD_INSTRUMENTS
Set the Estimated Rebate Value to the
SUM(I_FOOD_INSTRUMENTS_FOODS.QU
ANTITY *
F_REB_CONTRACTS.REBATE_FOR_UNIT).

If the I_BANK_REPORTS record is a Missing Issuance or was Issued and then subsequently Voided, then

Update F_OBLIGATIONS
Decrement the FI Obligated Count by 1

If the I_BANK_REPORTS.EST_REB_VALUE is not null, then

Decrement the Rebate Value FI Counter by 1.

Subtract the I_BANK_REPORTS.EST_REB_VALUE from the F_OBLIGATIONS.EST_REB_VALUE.

1.19.2.3 Outputs

See section 1.17.4.4

1.20 Search for potential dual enrollment clients

1.20.1 EC2_DUAL_ENROLL.SQL

Overview

This script is used to verify clients do not have dual enrollments. The script inserts records into the C_RESOLVED_CLIENTS table and C_RESOLUTIONS table for each potential dual enrollment candidate leaving the C_RESOLUTIONS.DATE_RESOLVED column blank to indicate that this potential dual enrollment should be investigated.

1.20.1.1 Inputs

EOD_CONTROLS
ENV_VARIABLES
C_RESOLVED_CLIENTS
C_FAMILY_ECONOMIC_UNITS
C_CLIENTS CC
C_CLIENTS CC1
C_CLIENTS CC2
C_RESOLUTIONS
E_C_CLIENTS
C_CLIENTS_DUAL

1.20.1.2 Selection

To retrieve the clients with potential dual enrollments, excluding compliance clients and clients that have already been resolved:

C_CLIENTS_DUAL CCD, E_C_CLIENTS ECC
WHERE ccd.client_id <> ecc.client_id
AND ccd.lasst_first_bdate <> ec.last_first_bdate
AND not exists (ccd.client_id, ecc.client_id)
In (slect cc.client_id, crc.client_id from c_resolutions)

Select more details of client from
C_clients & c_family_economic_units table

1.20.1.3 Processing

For each potential dual enrollment client found Loop

Insert a record into c_resolved_clients setting the columns as follows:

CLIENT_ID	= ECC.CLIENT_ID
CAT_CATEGORY_CODE	= C_CLIENTS.CAT_CATEGORY_CODE
OU_SEQ_ID	= CFEU.OU_SEQ_ID
LAST_NAME	= C_CLIENTS.LAST_NAME
FIRST_NAME	= C_CLIENTS.FIRST_NAME
BIRTH_DATE	= C_CLIENTS.BIRTH_DATE
GENDER	= C_CLIENTS.GENDER
DATE_CREATED	= EOD_CONTROLS.TO_DATE
CREATED_BY	= ENV_VARIABLES.ENV_VALUE where code = 'EOD_BY'
MI1	= C_CLIENTS.MI1
MI2	= C_CLIENTS.MI2

Insert a record into c_resolutions setting the columns as follows:

CRC_CLIENT_ID	= ECC.CLIENT_ID
CC_CLIENT_ID	= CCD.CLIENT_ID
DATE_CREATED	= EOD_CONTROLS.TO_DATE
CREATED_BY	= ENV_VARIABLES.ENV_VALUE where code = 'EOD_BY'

End Loop

1.20.1.4 Outputs

C_Resolutions
C_Resolved_clients

1.21 Compile new issuance and void information to send to bank

1.21.1 EC2_ ISSUE_FSMC.SQL

Overview

This script creates a text file of food instruments that should be sent to the bank.

1.21.1.1 Inputs

I_bank_reports ibr
O_organizational_units ou1
O_organizational_units ou2

1.21.1.2 Selection

I_BANK_REPORTS where send_code = 'B' (The FI has not been sent to the Bank yet)
And ((idis_disposition_code = 2 (Issued)
And issue_method in ('B', 'C', 'O', 'R', 'P')) (Batch, Compliance, On-Demand, Re-issued or Replaced)
OR idis_disposition_code = 3 (Voided)
OR idis_disposition_code = '5' and revalidation_code = '2')

1.21.1.3 Processing

spool issuean.dat

```
select  substr (ibr.serial_number, 1, 10)
                select substr(ibr.serial_number,1,10)
                 ,lpad(nvl(substr(ifit_fi_type_code,1,6),0),6,0)
                 ,rpad(nvl(substr(ifit_fi_type_code,7,2),'XX'),2,'XX')
                 ,'**'
                 ,rpad(ou2.org_code,2,' ')
                 ,' '
                 ,'0000000000'
                 ,rpad(to_char(nvl(ibr.issue_date,sysdate),'mmddyyyy'),8,'0')
                --      the comment below explains the decode mess that follows it.  Kevin M
11/24/2000
                --      if disposition = 2 (for issued) then
                --              if issue-method = 'O' or 'B' (for On-demand or batch) then
                --                      return 'D'
```

```
--                  elsif issue-method = 'C' (for Compliance-buy) then
--                          return 'C'
--                  elsif issue-method = 'R' (for Reissued checks) then
--                          return 'A'
--                  elsif issue-method = 'P' (for Replacement checks) then
--                          return 'A'
--          elsif disposition = 3 (for voided) then
--                  if void reason = 'B' (for stolen) then
--                          return 'S'
--                  else return 'V'
--          elsif disposition = 5 (for rejected) then
--                  if revalidation code = 2 then
--                          return 'O' (over-ride)
--          else return ' '
 ,rpad(nvl(decode(ibr.idis_disposition_code,'2',decode(ibr.issue_method,'O','D','B','D',
                                                       'C','C','R','A','P','A')
                   ,'3',decode(ibr.ivr_void_reason_code,'B','S','V')
                   ,'5',     decode(ibr.revalidation_code,'2','O')),' '),1,' ')
 ,rpad(nvl(decode(ibr.idis_disposition_code,'3',ibr.ivr_void_reason_code),' '),2,' ')
 ,rpad(to_char(nvl(ibr.first_date_to_use,sysdate),'mmddyyyy'),8,'0')
 ,rpad(to_char(nvl(ibr.last_date_to_use,sysdate),'mmddyyyy'),8,'0')
 ,substr(lpad(ltrim(to_char(nvl(ibr.maximum_amt,0),'9999.99')),7,'0'),1,4)
 ,substr(lpad(ltrim(to_char(nvl(ibr.maximum_amt,0),'9999.99')),7,'0'),6,2)
 ,'00000000000'
from
  i_bank_reports ibr
 ,o_organizational_units ou1
 ,o_organizational_units ou2
where   ibr.send_code = 'B'
and     ibr.ou_seq_id = ou1.seq_id
and     NVL(ou1.ou_seq_id,10) = ou2.seq_id
and     ((ibr.idis_disposition_code = '2'
        and ibr.issue_method in ('B','C','O','R','P'))
        or  ibr.idis_disposition_code = '3'
        or  (ibr.idis_disposition_code = '5' and ibr.revalidation_code = '2'));

select 'TOTAL',lpad(ltrim(to_char(count(*))),6,'0')
from
 i_bank_reports ibr
 ,o_organizational_units ou1
 ,o_organizational_units ou2
where   ibr.send_code = 'B'
and     ibr.ou_seq_id = ou1.seq_id
and     NVL(ou1.ou_seq_id,10) = ou2.seq_id
and     ((ibr.idis_disposition_code = '2'
        and ibr.issue_method in ('B','C','O','R','P'))
        or  ibr.idis_disposition_code = '3'
        or  (ibr.idis_disposition_code = '5' and ibr.revalidation_code = '2'));
```

1.21.1.4 Outputs

issuean.dat

1.22 EC4.BAT

Overview

This batch file is called by EC2.BAT and is part of the Agency upload process.

Processing

This script first reads the EC4.INI file to initialize and set temporary environmental variables.

The script then runs the executable file FTPTRANS.EXE which is a visual basic script that does all of the bank interface processing and does the following:

1. Renames the 'issuean.dat' file in the 'dump' directory to 'isanmmdd.dat'.
2. Logs into the web site and puts them into the 'load' directory.
3. Copies the three sending files in the 'dump' directory to the web site and moves them to the 'processed' directory.
4. Creates the import_file.bat file consisting of sql-loads and a move command for all files that haven't previously processed (not in the 'processed' directory).
5. Executes the import_file.bat script.

Then runs the following SQL scripts:

EC4_BANK.SQL
EC4.SQL

EC4.SQL updates ENV_VARIABLES and EOD_CONTROLS and then runs the following scripts:

EC4_CASHFLOW_UPD.SQL
EC4_MON_END.SQL
EC4_VEN_RISK_HISTORY.SQL
EC4_VEN_PEER_FI.SQL
EC4_VEN_PRICE.SQL
EC4_VEN_LOC_AGCY_FI.SQL
EC4_C1_SER_REP.SQL
EC4_SYCN_STATE_CLIENTS.SQL

For each agency; searches for %.flg (wildcard represents the Agency Id) to see if the Agency for exists, returns 'OK', and then deletes the .flg file.

Then runs the following SQL scripts:

EC4_STATE_CLIENTS.SQL
EC4_TRNC.SQL
EC4_AGCY_BEING_PROCESSED.SQL
EC4_RETRIEVE.SQL

EC4_TAB_DOWN.SQL
EC4_STATE_CLIENTS.SQL
EC4_GET_UPDATES.SQL

Outputs

See Appendix B.

1.23 Create consolidated record of all FI issuance and redemption information and create vendor payment records

1.23.1 EC4_BANK.SQL

1.23.1.1 Inputs

n/a

1.23.1.2 Selection

n/a

1.23.1.3 Processing

Executes the ec4_bnk_post.sql script

1.23.1.4 Outputs

n/a

1.23.2 EC4_BNK_POST.SQL

Overview

This script creates consolidated food instrument issuance and redemption information and creates vendor payment records.

The bank data Status values translate into i_food_instruments.idis_disposition_code values:
‘P’, ‘M’, and ‘S’ translate into i_food_instruments.idis_disposition_code = ‘4'
‘R’ translates into i_food_instruments.idis_disposition_code = ‘5'

The bank data Accept Code values are copied directly into i_food_instruments.bd_bank_disposition_code.

If the food instrument is stale dated but cleared by the bank then this script clears out the stale dated void information from that food instrument.

1.23.2.1 Inputs

E_BANK_DATAS
I_FOOD_INSTRUMENTS
F_OBLIGATIONS
F_OUTLAYS
I_FOOD_INSTRUMENT_FOODS
F_REB_CONTRACTS
F_REB_DETAILS
V_VENDORS
O_ORGANIZATIONAL_UNITS
I_DISPOSITIONS

1.23.2.2 Selection

For IDIS_DISPOSITION_CODE, I_FOOD_INSTRUMENTS.EST_REB_VALUE, REDEMPTION_AMT:
I_FOOD_INSTRUMENTS where the serial number equals the serial number from E_BANK_DATAS.

For FI_OBLIG_COUNT, REB_FI_COUNTER, EST_REB_VALUE
F_OBLIGATIONS where F_OBLIGATIONS FI type code and sequence id equal the food instruments FI type code and sequence id and the F_OBLIGATIONS category code and priority equal the food instruments category code and priority decoded from the certifications table for the specified time period.

For REDEEMED_FI_COUNT, REDEEMED_FI_VALUE:
F_OUTLAYS where F_OUTLAYS and F_OBLIGATIONS have equivalent values for category code, FI type, sequence id, priority, and fiscal month and year.

1.23.2.3 Processing

Read all reords returned from the bank: bank_data_rec

Find the i_food_instruments record whose serial_number matches the bank_data_rec.serial_number.

If the i_food_instruments.process_date is not null and i_food_instrument.redemption_amt is not null or bank_data_rec.status = 'R'

Insert a record into i_bank_datas with the i_food_instruments data and set
Post_code = 'N'
Send_code = 'N'
Message = 'rejected - duplicate redeemed check from bank'
End

Find the v_vendors.id from bank_data_rec.vendor_id

If bank_data_rec has a null process_date, amount, or status, write an error message and do not process the record.

Update I_Food_Instruments with the following information received from the bank_data_rec, received from the bank:

If the bank_data_rec.status is 'P', 'M', or 'S' (paid, manual, skeleton),
idis_disposition_code := 4 (indicating "redeemed")
redemption_amt := bank_data_rec.amount / 100
cleared_date := bank_data_rec.process_date
reject_date := null
requested_amt := null
else
idis_disposition_code := 5 (indicating "rejected")
redemption_amt := null
cleared_date := null
reject_date := bank_data_rec.process_date
requested_amt := bank_data_rec.amount / 100
end if
process_date := bank_data_rec.process_date
obligated_amt := null
bd_bank_disposition_code := bank_data_rec.accept_code

If the food instrument is stale dated but cleared by the bank then this script clears out the stale dated void information from that food instrument.

If i_food_instruments.idis_disposition_code = '3',

Generate l_fiscal_year and l_fiscal_month from void_date
else
If idis_disposition_code = '3' (redeemed)
Generate l_fiscal_year and l_fiscal_month from redeemed_date

if i_food_instruments.idis_disposition_code = '4' (redeemed), perform five steps:

1. Reduce obligations:
Get f_obligations record for the i_food_instruments.ifit_fi_type,
i_food_instrument_types.ou_seq_id and
f_obligations.cat_category_code equal to the
category from the client using the fi,
where the FI prescription matches the FI
and the f_obligations fiscal year and
month.
Update f_obligations with fi_oblig_count = fi_oblig_count - 1
if i_food_instruments.est_reb_value is not null,
Update f_obligations with reb_fi_counter = reb_fi_counter = 1

2. Increase Outlays:
Get f_outlays record that matches the f_obligations record for l_fiscal_year and l_fiscal_month

if not found,
Create a record to match the f_obligations with l_fiscal_year and l_fiscal_month
Get the newly created record

Update f_outlays, incrementing redeemed_fi_count and adding
i_food_instrument.redemption_amount to redeemed_fi_value

3. Increase f_reb_details quantity and value:
Search for f_reb_contracts records for the i_food_intrument_foods that make up
the i_food_instruments.ifit_fi_type_code where the cleared_date is
between the f_reb_contracts.start_date, end_date
Set local variable l_total_rebates = sum(f_reb_contracts.rebate_per_unit *
i_food_instrument_foods.quantity) where the cleared_date is between the
f_reb_contracts.start_date, end_date

If l_total_rebates is not null then (rebate information for the food instrument was found)

Find the f_reb_details records
where f_reb_contracts.if_food_id matches the i_food_instrument_foods.if_food_id
for the i_food_instruments.ifit_fi_type_code and the
i_food_intruments.cleared_date is between
f_reb_contracts.start_date and end_date

Set local variables:
l_unit_quantity = f_reb_details.unit_quantity +
f_reb_details.quantity
l_value_for_units = f_reb_details.value_for_units +
(f_reb_details.unit_quantity *
f_reb_details.quantity)

if f_reb_details.replacement_flag is null,
create the f_reb_details record:
fol_activity_year = l_fiscal_year
fol_activity_month = l_fiscal_month
if i_food_instrument.issue_method = 'R'
Replacement_flag = 'Y'
Else
Replacement_flag = 'N'
Else
Update the f_reb_details record with:
Unit_quantity = l_unit_quantity
Value_for_units = l_value_for_units
if i_food_instrument.issue_method = 'R'
Replacement_flag = 'Y'
Else
Replacement_flag = 'N'
End;

4. Delete reject reasons for revalidated FI's

If the bank_data_rec.accept_code = 'U' (record was revalidated by the bank),
delete from i_fi_reject_reasons for the bank_data_rec.serial_number

5. Insert a record into i_bank_data with the i_food_instruments data and set
Post_code = 'Y'
Send_code = 'L'
Message = 'This check received from bank'

If the record is missing issuance, then create the I_Food_Instrument record:

Insert i_food_instruments, setting the following values:
serial_number = bank_data_rec.serial_number
ou_seq_id = o_organizational_units.seq_id for the State Agency
compliance_buy_flag = 'N'
idis_disposition_code = decode (bank_data_rec.status, 'R',5,'P',4,'M',4,'S',4)
bd_bank_disposition_code = bank_data_rec.accept_code

ven_id = v_vendors.id where bank_data_rec.vendor_number = v_vendors.vendor_id

if bank_data_rec.status = 'R'
requested_amt := bank_data_rec.amount / 100
reject_date := bank_data_rec.process_date
redemption_amt = null
cleared_date = ' '
else
redemption_amt := bank_data_rec.amount / 100
cleared_date = bank_data_rec.process_date
requested_amt = null
reject_date = null

missing_issuance_flag = 'N'
process_date = bank_data_rec.process_date

Insert a record into i_bank_datas with the i_food_instruments data and set
Post_code = 'Y'
Send_code = 'N'
Message = 'This is missing issuance check'

Add rejection reasons:

if length(bank_data_rec.rejection_code) <> 0
For each character in the rejection_code (for i = 1 to length(bank_data_rec.rejection_code))
insert into i_fi_reject_reasons, setting:
ifi_serial_number = bank_data_rec.serial_number
irr_reject_reason_code = substr(bank_data_rec.rejection_code,i,1)
End;

1.23.2.4 Outputs

I_Food_Instruments
I_Bank_Reports
F_Obligations
F_Outlays
F_Reb_Details
I_FI_Reject_Reasons

Failure messages for:
-Obligations table
-deleting reject reasons records
-creating records in I_FOOD_INSTRUMENTS
-creating I_FI_REJECT_REASONS

1.23.3 EC4.SQL

Overview

Runs SQL scripts for EC4.BAT.

1.23.3.1 Input

eod_controls
env_variables

1.23.3.2 Selection

env_variables
where code = 'EOD_BY'

1.23.3.3 Processing

update eod_controls, setting TO_DATE to the system date
update env_variables, setting env_value to 'EC4' where code = 'EOD_BY'

run SQL script EC4_CASHFLOW_UPD
run SQL script EC4_MON_END
run SQL script EC4_VEN_PEER_FI
run SQL script EC4_VEN_LOC_AGCY_FI
run SQL script EC4_VEN_PRICE

update eod_controls, setting FROM_DATE = TO_DATE, GOOD_PROC_DATE = TO_DATE

1.23.3.4 Outputs

end_controls
env_varibles
Log file: EC4_SQL.LOG

1.24 Create date of current food funds available from all sources as well as current food obligations and expenditures to establish current cash position

1.24.1 EC4_CASHFLOW_UPD.SQL

Overview

This EOD script is used to update summary information in the caseloads table for each distinct month/year where data has been changed in obligations and outlays.

1.24.1.1 Inputs

ENV_VARIABLES
EOD_CONTROLS
F_OBLIGATIONS
F_ANNUAL_FACTORS
F_OUTLAYS
I_FOOD_INSTRUMENTS
C_FOOD_PACKAGE_PRESCRIPTIONS
C_CERTIFICATIONS
F_REB_DETAILS
F_BUDGETS

1.24.1.2 Processing

The F_CASHFLOWS table is populated by summarizing information from other Financial tables for the F_CASHFLOWS.ACTIVITY_MONTH and F_CASHFLOWS.ACTIVITY_YEAR. The F_CASHFLOWS.A_NADJ_* represents unadjusted obligations and outlays, and is used when the F_ANNUAL_FACTORS.REDEEM_RATE_METHOD = '3'. The F_CASHFLOWS.A_NADJ_PRERB_OBLIG_FROM is set to the SUM(F_OBLIGATIONS.FI_OBLIG_COUNT x F_OBLIGATIONS.EST_PER_FI_VALUE) for the Fiscal Month and Year. If the Fiscal Month is 2 or more months before the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_NADJ_OBLIG_FOR_MON0 and F_CASHFLOWS.A_NADJ_OBLIG_FOR_MON1 are both set to 0. F_CASHFLOWS.A_NADJ_OBLIG_FOR_MON2 is set to F_CASHFLOWS.A_NADJ_PRERB_OBLIG_FROM. If the Fiscal Month is 1 month before the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_NADJ_OBLIG_FOR_MON0 is set to 0. (F_BUDGETS.EST_FI_MON1_FACTOR + F_BUDGETS.EST_FI_MON2_FACTOR) is stored as a variable V_FACTORS_TOTAL. F_CASHFLOWS.A_NADJ_OBLIG_FOR_MON1 is set to (F_CASHFLOWS.A_NADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON1_FACTOR / V_FACTORS_TOTAL). F_CASHFLOWS.A_NADJ_OBLIG_FOR_MON2 is set to (F_CASHFLOWS.A_NADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON2_FACTOR / V_FACTORS_TOTAL). If the Fiscal Month is greater than or equal to the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_NADJ_OBLIG_FOR_MON0 is set to (F_CASHFLOWS.A_NADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON0_FACTOR). F_CASHFLOWS.A_NADJ_OBLIG_FOR_MON1 is set to (F_CASHFLOWS.A_NADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON1_FACTOR). F_CASHFLOWS.A_NADJ_OBLIG_FOR_MON2

is set to (F_CASHFLOWS.A_NADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON2_FACTOR).

The F_CASHFLOWS.A_NADJ_REB_OBLIG_FROM is set to the SUM(F_OBLIGATIONS.EST_REB_VALUE) for the Fiscal Month and Year. If the Fiscal Month is 2 or more months before the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_NADJ_REB_FOR_MON0 and F_CASHFLOWS.A_NADJ_REB_FOR_MON1 are both set to 0. F_CASHFLOWS.A_NADJ_REB_FOR_MON2 is set to F_CASHFLOWS.A_NADJ_REB_OBLIG_FROM. If the Fiscal Month is 1 month before the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_NADJ_REB_FOR_MON0 is set to 0. (F_BUDGETS.EST_FI_MON1_FACTOR + F_BUDGETS.EST_FI_MON2_FACTOR) is stored as a variable V_FACTORS_TOTAL. F_CASHFLOWS.A_NADJ_REB_FOR_MON1 is set to (F_CASHFLOWS.A_NADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON1_FACTOR / V_FACTORS_TOTAL). F_CASHFLOWS.A_NADJ_REB_FOR_MON2 is set to (F_CASHFLOWS.A_NADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON2_FACTOR / V_FACTORS_TOTAL). If the Fiscal Month is greater than or equal to the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_NADJ_REB_FOR_MON0 is set to (F_CASHFLOWS.A_NADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON0_FACTOR). F_CASHFLOWS.A_NADJ_REB_FOR_MON1 is set to (F_CASHFLOWS.A_NADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON1_FACTOR). F_CASHFLOWS.A_NADJ_REB_FOR_MON2 is set to (F_CASHFLOWS.A_NADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON2_FACTOR).

The F_CASHFLOWS.A_FADJ_* represent obligations adjusted for redemption probability by FI Type, and is used when the F_ANNUAL_FACTORS.REDEEM_RATE_METHOD = '1'. The F_CASHFLOWS.A_FADJ_PRERB_OBLIG_FROM is set to the SUM(F_OBLIGATIONS.FI_OBLIG_COUNT x F_OBLIGATIONS.ADJ_PER_FI_VALUE) for the Fiscal Month and Year. If the Fiscal Month is 2 or more months before the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_FADJ_OBLIG_FOR_MON0 and F_CASHFLOWS.A_FADJ_OBLIG_FOR_MON1 are both set to 0. F_CASHFLOWS.A_FADJ_OBLIG_FOR_MON2 is set to F_CASHFLOWS.A_FADJ_PRERB_OBLIG_FROM. If the Fiscal Month is 1 month before the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_FADJ_OBLIG_FOR_MON0 is set to 0. (F_BUDGETS.EST_FI_MON1_FACTOR + F_BUDGETS.EST_FI_MON2_FACTOR) is stored as a variable V_FACTORS_TOTAL. F_CASHFLOWS.A_FADJ_OBLIG_FOR_MON1 is set to (F_CASHFLOWS.A_FADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON1_FACTOR / V_FACTORS_TOTAL). F_CASHFLOWS.A_FADJ_OBLIG_FOR_MON2 is set to (F_CASHFLOWS.A_FADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON2_FACTOR / V_FACTORS_TOTAL). If the Fiscal Month is greater than or equal to the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_FADJ_OBLIG_FOR_MON0 is set to (F_CASHFLOWS.A_FADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON0_FACTOR). F_CASHFLOWS.A_FADJ_OBLIG_FOR_MON1 is set to (F_CASHFLOWS.A_FADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON1_FACTOR). F_CASHFLOWS.A_FADJ_OBLIG_FOR_MON2 is set to (F_CASHFLOWS.A_FADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON2_FACTOR).

The F_CASHFLOWS.A_FADJ_REB_OBLIG_FROM is set to the SUM(F_OBLIGATIONS.EST_REB_VALUE x F_OBLIGATIONS.EST_REDEEM_FACTOR) for the Fiscal Month and Year. If the Fiscal Month is 2 or more months before the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_FADJ_REB_FOR_MON0 and F_CASHFLOWS.A_FADJ_REB_FOR_MON1 are both set to 0. F_CASHFLOWS.A_FADJ_REB_FOR_MON2 is set to F_CASHFLOWS.A_FADJ_REB_OBLIG_FROM. If the Fiscal Month is 1 month before the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_FADJ_REB_FOR_MON0 is set to 0. (F_BUDGETS.EST_FI_MON1_FACTOR + F_BUDGETS.EST_FI_MON2_FACTOR) is stored as a variable V_FACTORS_TOTAL. F_CASHFLOWS.A_FADJ_REB_FOR_MON1 is set to (F_CASHFLOWS.A_FADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON1_FACTOR / V_FACTORS_TOTAL). F_CASHFLOWS.A_FADJ_REB_FOR_MON2 is set to (F_CASHFLOWS.A_FADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON2_FACTOR / V_FACTORS_TOTAL). If the Fiscal Month is greater than or equal to the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_FADJ_REB_FOR_MON0 is set to (F_CASHFLOWS.A_FADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON0_FACTOR). F_CASHFLOWS.A_FADJ_REB_FOR_MON1 is set to (F_CASHFLOWS.A_FADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON1_FACTOR). F_CASHFLOWS.A_FADJ_REB_FOR_MON2 is set to (F_CASHFLOWS.A_FADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON2_FACTOR).

The F_CASHFLOWS.A_TADJ_* represent obligations adjusted by a typical redemption rate, and is used when the F_ANNUAL_FACTORS.REDEEM_RATE_METHOD = '2'. The F_CASHFLOWS.A_TADJ_PRERB_OBLIG_FROM is set to the (SUM(F_OBLIGATIONS.FI_OBLIG_COUNT x F_OBLIGATIONS.EST_PER_FI_VALUE) x SUM(F_OBLIGATIONS.FI_OBLIG_COUNT x F_OBLIGATIONS.EST_REDEEM_FACTOR) / SUM(F_OBLIGATIONS.FI_OBLIG_COUNT)) for the Fiscal Month and Year. If the Fiscal Month is 2 or more months before the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_TADJ_OBLIG_FOR_MON0 and F_CASHFLOWS.A_TADJ_OBLIG_FOR_MON1 are both set to 0. F_CASHFLOWS.A_TADJ_OBLIG_FOR_MON2 is set to F_CASHFLOWS.A_TADJ_PRERB_OBLIG_FROM. If the Fiscal Month is 1 month before the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_TADJ_OBLIG_FOR_MON0 is set to 0. (F_BUDGETS.EST_FI_MON1_FACTOR + F_BUDGETS.EST_FI_MON2_FACTOR) is stored as a variable V_FACTORS_TOTAL. F_CASHFLOWS.A_TADJ_OBLIG_FOR_MON1 is set to (F_CASHFLOWS.A_TADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON1_FACTOR / V_FACTORS_TOTAL). F_CASHFLOWS.A_TADJ_OBLIG_FOR_MON2 is set to (F_CASHFLOWS.A_TADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON2_FACTOR / V_FACTORS_TOTAL). If the Fiscal Month is greater than or equal to the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_TADJ_OBLIG_FOR_MON0 is set to (F_CASHFLOWS.A_TADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON0_FACTOR). F_CASHFLOWS.A_TADJ_OBLIG_FOR_MON1 is set to (F_CASHFLOWS.A_TADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON1_FACTOR). F_CASHFLOWS.A_TADJ_OBLIG_FOR_MON2 is set to (F_CASHFLOWS.A_TADJ_PRERB_OBLIG_FROM x F_BUDGETS.EST_FI_MON2_FACTOR).

The F_CASHFLOWS.A_TADJ_REB_OBLIG_FROM is set to the ((SUM(F_OBLIGATIONS.EST_REB_VALUE) /

SUM(F_OBLIGATIONS.REB_FI_COUNTER)) x
(SUM(F_OBLIGATIONS.FI_OBLIG_COUNT x F_OBLIGATIONS.EST_REDEEM_RATE) / SUM(F_OBLIGATIONS.FI_OBLIG_COUNT))) for the Fiscal Month and Year. If the Fiscal Month is 2 or more months before the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_TADJ_REB_FOR_MON0 and F_CASHFLOWS.A_TADJ_REB_FOR_MON1 are both set to 0. F_CASHFLOWS.A_TADJ_REB_FOR_MON2 is set to F_CASHFLOWS.A_TADJ_REB_OBLIG_FROM. If the Fiscal Month is 1 month before the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_TADJ_REB_FOR_MON0 is set to 0. (F_BUDGETS.EST_FI_MON1_FACTOR + F_BUDGETS.EST_FI_MON2_FACTOR) is stored as a variable V_FACTORS_TOTAL. F_CASHFLOWS.A_TADJ_REB_FOR_MON1 is set to (F_CASHFLOWS.A_TADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON1_FACTOR / V_FACTORS_TOTAL). F_CASHFLOWS.A_TADJ_REB_FOR_MON2 is set to (F_CASHFLOWS.A_TADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON2_FACTOR / V_FACTORS_TOTAL). If the Fiscal Month is greater then or equal to the (Fiscal Month of SYSDATE), then F_CASHFLOWS.A_TADJ_REB_FOR_MON0 is set to (F_CASHFLOWS.A_TADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON0_FACTOR). F_CASHFLOWS.A_TADJ_REB_FOR_MON1 is set to (F_CASHFLOWS.A_TADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON1_FACTOR). F_CASHFLOWS.A_TADJ_REB_FOR_MON2 is set to (F_CASHFLOWS.A_TADJ_REB_OBLIG_FROM x F_BUDGETS.EST_FI_MON2_FACTOR).

F_CASHFLOWS.A_NADJ_OBLIG_IN_ACTIV_MON is set to the sum of
F_CASHFLOWS.A_NADJ_OBLIG_FOR_MON0 +
F_CASHFLOWS.A_NADJ_OBLIG_FOR_MON1 +
F_CASHFLOWS.A_NADJ_OBLIG_FOR_MON2.
F_CASHFLOWS.A_FADJ_OBLIG_IN_ACTIV_MON is set to the sum of
F_CASHFLOWS.A_FADJ_OBLIG_FOR_MON0 +
F_CASHFLOWS.A_FADJ_OBLIG_FOR_MON1 +
F_CASHFLOWS.A_FADJ_OBLIG_FOR_MON2.
F_CASHFLOWS.A_TADJ_OBLIG_IN_ACTIV_MON is set to the sum of
F_CASHFLOWS.A_TADJ_OBLIG_FOR_MON0 +
F_CASHFLOWS.A_TADJ_OBLIG_FOR_MON1 +
F_CASHFLOWS.A_TADJ_OBLIG_FOR_MON2.

F_CASHFLOWS.A_NADJ_REB_IN_ACTIV_MON is set to the sum of
F_CASHFLOWS.A_NADJ_REB_FOR_MON0 +
F_CASHFLOWS.A_NADJ_REB_FOR_MON1 +
F_CASHFLOWS.A_NADJ_REB_FOR_MON2.
F_CASHFLOWS.A_FADJ_REB_IN_ACTIV_MON is set to the sum of
F_CASHFLOWS.A_FADJ_REB_FOR_MON0 +
F_CASHFLOWS.A_FADJ_REB_FOR_MON1 +
F_CASHFLOWS.A_FADJ_REB_FOR_MON2.
F_CASHFLOWS.A_TADJ_REB_IN_ACTIV_MON is set to the sum of
F_CASHFLOWS.A_TADJ_REB_FOR_MON0 +
F_CASHFLOWS.A_TADJ_REB_FOR_MON1 +
F_CASHFLOWS.A_TADJ_REB_FOR_MON2.

The actual pre-rebate outlays are stored in F_CASHFLOWS.A_PRERB_OUTLAY_FROM, which is set to SUM(F_OUTLAYS.REDEEMED_FI_VALUE) for the Fiscal Month and Year. F_CASHFLOWS.A_OUTLAY_FOR_MON0 is set to

SUM(F_OUTLAYS.REDEEMED_FI_VALUE) for the Fiscal Month and Year where the F_OUTLAYS.ACTIVTY_MONTH and F_OUTLAYS.ACTIVITY_YEAR are equal to the Fiscal Month and Year. F_CASHFLOWS.A_OUTLAY_FOR_MON1 is set to SUM(F_OUTLAYS.REDEEMED_FI_VALUE) for the Fiscal Month and Year where the F_OUTLAYS.ACTIVTY_MONTH and F_OUTLAYS.ACTIVITY_YEAR are equal to the previous Fiscal Month and Year. F_CASHFLOWS.A_OUTLAY_FOR_MON2 is set to SUM(F_OUTLAYS.REDEEMED_FI_VALUE) for the Fiscal Month and Year where the F_OUTLAYS.ACTIVTY_MONTH and F_OUTLAYS.ACTIVITY_YEAR are equal to the Fiscal Month and Year of two months ago. F_CASHFLOWS.A_OUTLAY_IN_ACTIV_MON is set to the SUM(F_OUTLAYS.REDEEMED_FI_VALUE) for all Fiscal Months and Years where F_OUTLAYS.ACTIVITY_MONTH and F_OUTLAYS.ACTIVITY_YEAR are equal to the F_CASHFLOWS.ACTIVTY_MONTH and F_CASHFLOWS.ACTIVITY_YEAR.

The actual billable rebates are stored in F_CASHFLOWS.A_OUTLAY_REB_FROM, which is set to SUM(F_REB_DETAILS.VALUE_FOR_UNITS) for the Fiscal Month and Year. F_CASHFLOWS.A_OUTLAY_REB_FOR_MON0 is set to SUM(F_REB_DETAILS.VALUE_FOR_UNITS) for the Fiscal Month and Year where the F_REB_DETAILS.FOL_ACTIVTY_MONTH and F_REB_DETAILS.FOL_ACTIVITY_YEAR are equal to the Fiscal Month and Year. F_CASHFLOWS.A_OUTLAY_REB_FOR_MON1 is set to SUM(F_REB_DETAILS.VALUE_FOR_UNITS) for the Fiscal Month and Year where the F_REB_DETAILS.FOL_ACTIVTY_MONTH and F_REB_DETAILS.FOL_ACTIVITY_YEAR are equal to the previous Fiscal Month and Year.
F_CASHFLOWS.A_OUTLAY_REB_FOR_MON2 is set to SUM(F_REB_DETAILS.VALUE_FOR_UNITS) for the Fiscal Month and Year where the F_REB_DETAILS.FOL_ACTIVTY_MONTH and F_REB_DETAILS.FOL_ACTIVITY_YEAR are equal to the Fiscal Month and Year of two months ago.
F_CASHFLOWS.A_OUTLAY_REB_IN_ACTIV_MON is set to the SUM(F_REB_DETAILS.VALUE_FOR_UNITS) for all Fiscal Months and Years where F_REB_DETAILS.FOL_ACTIVTY_MONTH and F_REB_DETAILS.FOL_ACTIVITY_YEAR are equal to the F_CASHFLOWS.ACTIVTY_MONTH and F_CASHFLOWS.ACTIVITY_YEAR.

Both F_BUDGETS.ACT_PARTICIP and F_CASHFLOWS.A_REDEEM_PARTICIP are generated from SUM(F_CASELOADS.CLIENT_COUNT) for the Fiscal Month and Year where F_CASELOAD_TYPES.DESCRIPTION = 'PARTICIPANT'. The F_MONTHLY_CATEGORY_FACTORS.ACT_PARTICIPANT is populated with the SUM(F_CASELOADS.CLIENT_COUNT) for the Fiscal Month and Year, and Category where F_CASELOAD_TYPES.DESCRIPTION = 'PARTICIPANT'.

1.24.1.3 Outputs

States when processing begins and ends for obligations and outlays.

F_Budgets
F_Cashflows
F_Monthly_Category_Factors

1.25 Financial (end of month) - Populate F_INCOME to POVERTY table. These values are used in populating of the caseload table.

1.25.1 EC4_MON_END.SQL

Overview

This end of day script is used to perform population of income and poverty tables at the end of month. It will insert the junction of the C_INCOME_LEVELS and the F_POVERTY_LEVELS tables into the F_INCOMES_TO_POVERTIES table. It will update the caseload type details table with latest available data and create a new set of records in the caseloads table for the following month.

1.25.1.1 Inputs

eod_controls
c_income_levels
f_poverty_levels
f_poverty_basis

1.25.1.2 Processing

If the SYSDATE is the after the last day of the month EOD_CONTROLS.IPL_DATE, then
Select the max(F_POVERTY_LEVELS.BEGIN_DATE), the max(F_POVERTY_BASES.BEGIN_DATE), and the max(C_INCOME_LEVELS.BEGIN_DATE) that is on or before the first date of the month after the EOD_CONTROLS.IPL_DATE.
If any of these three dates are the first date of the month after the EOD_CONTROLS.IPL_DATE, then
For all C_INCOME_LEVELS records for the Begin Date,
Insert into F_INCOMES_TO_POVERTIES
The foreign keys are derived from the primary keys of the F_POVERTY_LEVELS and C_INCOME_LEVELS records

The Begin Date is derived from the C_INCOME_LEVELS.BEGIN_DATE

The Income High is derived from the F_POVERTY_LEVELS.HIGH_PERCENTAGE * C_INCOME_LEVELS.INCOME_HIGH / (F_POVERTY_BASES.POVERTY_BASIS * 100)

The Income Low is derived from the F_POVERTY_LEVELS.LOW_PERCENTAGE *

C_INCOME_LEVELS.INCOME_HIGH /
(F_POVERTY_BASES.POVERTY_BASIS * 100)

Update EOD_CONTROLS
Set the IPL_DATE to the month after the current EOD_CONTROLS.IPL_DATE

Set the MONTHLY_SW to 'Y'

If the max(F_INCOMES_TO_POVERTIES.BEGIN_DATE) equals EOD_CONTROLS.IPL_DATE, then
For F_INCOMES_TO_POVERTIES records where the Begin Date is in the month of EOD_CONTROLS.IPL_DATE
Update F_CASELOAD_TYPE_DETAILS where the Family Size, Income, and Fiscal Year and Month are equal to the F_INCOMES_TO_POVERTIES record
Set the Poverty Level foreign key to F_INCOMES_TO_POVERTIES.FPL_ID

For caseload processing please refere to Section 1.38

1.25.1.3 Outputs

eod_controls
f_incomes_to_poverties

1.26 Vendor (end of month) calculate and update peer group averages

1.26.1 EC4_VEN_PEER_FI.SQL

Overview

Updates or inserts into peer group averages for the food instruments disposition code of '4' (Redeemed). The following data elements are updated:

* Total redemption amount
* Average participants
* Average redemption amount
* Maximum redemption amount
* Total redeemed FI's
* Survey flag
* Date modified
* Modified by

1.26.1.1 Inputs

V_VENDORS
I_FOOD_INSTRUMENTS
V_PEER_GROUP_AVGS
V_PEER_GROUPS

1.26.1.2 Selection

To calculate V_PEER_GROUP_AVGS.AVG_PARTICIPANTS:

I_FOOD_INSTRUMENTS where I_FOOD_INSTRUMENTS.CLEARED DATE is in the past three months, retrieved separately
and I_FOOD_INSTRUMENTS.IDIS_DISPOSITION_CODE = 4 (Redeemed)
for the Vendor and Peer Group

For all other calculations:

I_FOOD_INSTRUMENTS where I_FOOD_INSTRUMENTS.CLEARED_DATE is within the last 30 calendar days
and I_FOOD_INSTRUMENTS.IDIS_DISPOSITION_CODE = 4 (Redeemed)
for the Vendor and Peer Group
grouped by V_VENDORS.PEE_PEER_GROUP,
I_FOOD_INSTRUMENTS.IFIT_FI_TYPE_CODE

1.26.1.3 Processing

end_date := today's date
start_date := end_date minus 30 days
For I_FOOD_INSTRUMENTS where cleared_date is between start_date and end_date and
idis_disposition_code = 4 (Redeemed)
grouped by V_VENDORS.PEE_PEER_GROUP,
I_FOOD_INSTRUMENTS.IFIT_FI_TYPE_CODE

fi_count := count(i_food_instruments.serial_number)
total_red_amt := sum(i_food_instruments.redemption_amt)

-- if there are not three data points, do not recalculate:
if count(I_FOOD_INSTRUMENTS.SERIAL_NUMBER) for the Peer Group and FI Type is < 3,
set the AVG_REDEMPTION_AMT to null
and set I_FOOD_INSTRUMENTS.RECALCULATE_FAILED = 'Y'

-- if recalculating would change the average by more than 15%, do not recalculate:
new_avg_redemption_amt := total_red_amt / fi_count

Else if abs ((i_food_instruments.avg_redemption_amt - new_avg_redemption_amt) /
i_food_instruments.avg_redemption_amt) > .15
set the AVG_REDEMPTION_AMT to null
and set I_FOOD_INSTRUMENTS.RECALCULATE_FAILED = 'Y'

-- otherwise, go ahead:
Else
set I_FOOD_INSTRUMENTS.RECALCULATE_FAILED = 'N'

-- count the average number of participants for the past three months
For I_FOOD_INSTRUMENTS where I_FOOD_INSTRUMENTS.CLEARED DATE is
in the past three months, retrieved separately by month, and
I_FOOD_INSTRUMENTS.IDIS_DISPOSITION_CODE = 4
for the Vendor and Peer Group

client_count := distinct count(I_FOOD_INSTRUMENTS.CC_CLIENT_ID),
averaged for the cleared_date in the past three months, for the Vendor
and Peer Group

-- find the percent override for the FI type, in case a record must be added to v_peer_group_avgs:
temp_percent := select default_percent from v_peer_groups for the peer group

-- update/insert into the v_peer_group_avgs table
if v_peer_group_avgs record exists already for the Peer Group/FI,

Update v_peer_group_avgs set
total_redemption_amt = total_red_amt,
avg_participants = client_count,
total_redeemed_fis = fi_count,
avg_redemption_amt = total_red_amt / fi_count,

max_redemption_amt = ((Total_Red_Amt/FI_Count)*
((100+PERCENT)/100)),
update_survey_flag = ‘Y’

Else

Insert into v_peer_group_avgs, setting
total_redemption_amt = total_red_amt,
avg_participants = client_count,
total_redeemed_fis = fi_count,
avg_redemption_amt = total_red_amt / fi_count,
percent = temp_percent,
max_redemption_amt = ((Total_Red_Amt/FI_Count)*
((100+temp_percent)/100)),
update_survey_flag = ‘Y’

1.26.1.4 Outputs

V_PEER_GROUP_AVGS
Printed log of start, insert/update, and end messages:

```
dbms_output.put_line('Starting Peer Group Average EOD');
dbms_output.put_line('Inserting Peer / FI :'||Temp_Peer_Group_ID||' / '||Fi_Type);
dbms_output.put_line('Updating Peer / FI :'||Temp_Peer_Group_ID||' / '||Fi_Type);
dbms_output.put_line('Peer Group Average Data Updated Successfully');
```

On failure:

```
dbms_output.put_line('Peer Group Average Table Update Failed');
```

See Appendix A for further information.

1.26.2 EC4_VEN_LOC_AGCY_FI

Overview

Updates or inserts into the local agency averages for the food instruments disposition code of '4' (Redeemed). The following data elements are updated:

* Total redemption amount
* Average redemption amount
* Total redeemed FI's
* Survey flag
* Date modified
* Modified by

1.26.2.1 Input

I_FOOD_INSTRUMENTS
O_ORGANIZATIONAL_UNITS
V_VENDORS

1.26.2.2 Selection

I_FOOD_INSTRUMENTS where I_FOOD_INSTRUMENTS.CLEARED_DATE is within the
last 30 calendar days
and I_FOOD_INSTRUMENTS.IDIS_DISPOSITION_CODE = 4
for the Vendor and Local Agency
grouped by V_VENDOR.OU_SEQ_ID_PRI_LA,
I_FOOD_INSTRUMENTS.IFIT_FI_TYPE_CODE

1.26.2.3 Processing

end_date := today's date
start_date := end_date minus 30 days
For I_FOOD_INSTRUMENTS where cleared_date is between start_date and end_date and
idis_disposition_code = 4 (redeemed)
grouped by the redeeming Vendor's local agency and the FI Type

fi_count := count(i_food_instruments.serial_number)
total_red_amt := sum(i_food_instruments.redemption_amt)

-- if there are not three data points, do not recalculate:
if count(I_FOOD_INSTRUMENTS.SERIAL_NUMBER) for the Local Agency and FI Type is <
3, set the AVG_REDEMPTION_AMT to null

-- if recalculating would change the average by more than 15%, do not recalculate:
new_avg_redemption_amt := total_red_amt / fi_count

ElseIf abs ((v_la_fi_avgs.avg_redemption_amt - new_avg_redemption_amt) /
v_la_fi_avgs.avg_redemption_amt) > .15
set the AVG_REDEMPTION_AMT to null

-- otherwise, go ahead:
-- update/insert into the V_LA_FI_AVGS table
else

if V_LA_AVGS record exists already for the Local Agency/FI,

Update V_LA_FI_AVGS set
total_redemption_amt = total_red_amt,
total_redeemed_fis = fi_count,
avg_redemption_amt = total_red_amt / fi_count,
update_survey_flag = 'Y'

Else

Insert into V_LA_FI_AVGS, setting
total_redemption_amt = total_red_amt,
total_redeemed_fis = fi_count,
avg_redemption_amt = total_red_amt / fi_count,
update_survey_flag = 'Y'

1.26.2.4 Output

V_LA_FI_AVGS
Printed log of start, insert/update, and end messages
dbms_output.put_line('Starting LA Average EOD');

```
        dbms_output.put_line('Inserting LA / FI :'||Temp_LA_ou_ID||' / '||Fi_Type);
        dbms_output.put_line('Updating Peer / FI :'||Temp_LA_ou_ID||' / '||Fi_Type);
        dbms_output.put_line('LA Average Data Updated Successfully');
On failure:
        dbms_output.put_line('LA Average Table Update Failed');
```

See Appendix A for further information

1.27 Vendor (end of month) run and print analysis factor reports

1.27.1 EC4_VEN_PRICE.SQL

Overview

Vendors purchasing analysis and ranking reports are produced from this script. The FI cost for Vendors is calculated and stored, based on their Food Price Surveys, and stored in V_FI_PRICES. Vendors are ranked by average maximum price, redemption value, and peer group identification.

A recalculation for a v_fi_prices record is performed if:

- the user checked a new compare flag on the Vendor Food Price Survey, Price Analysis, Compare/Risk window, or
- the user changed a Vendor survey price that is a component of an FI that has the compare flag checked on the Vendor Food Price Survey, Price Analysis, or Compare/Risk window. This would be indicated by the update_survey_flag set to 'Y' on the v_fi_prices table.

1.27.1.1 Inputs

i_food_instrument_foods
i_food_intruments
i_foods
v_fi_prices
v_peer_group_avgs
v_prices
v_vendors

1.27.1.2 Selection

1.27.1.3 Processing

Create a temporary table Temp_Ven_Price_Survey
Vendor ID
FI Type
Peer Group ID
FI Count
Avg Redemption Value
Redemption Rank *
Minimum Price

Maximum Price *
Rank *

Load the following data into the Temp_Ven_Price_Survey table without creating duplicate records:

a. Vendor/FI combinations that were checked on the Price Analysis Compare/Risk window since the last successful end of day run:

Select FI type from i_food_instrument_type where compare_flag = 'Y', for Vendors having any v_surveys record with rpt_rcvd_date or rpt_postmarked_date not null, and rep_req_review = 'N'. Include the Peer Group ID for the Vendor.

b. Vendors/FI combinations whose survey prices changed:

Select Vendor Id, FI Type from v_fi_prices where update_survey_flag = 'Y'. Include the Peer Group ID for the Vendor.

1. For each Vendor/FI combination on the Temp_Ven_Price_Survey table, calculate and store the maximum price for the FI's, based on Vendors' Price Survey:

Set local variables:
d_ifit_fi_type_code (FI Type)
d_pee_peer_group_id (peer group)

For each food in the FI Type, set local variables:

d_if_food_id := if_food_id (Food ID)
d_quantity := quantity (of the food in the FI)
d_unit_size := unit_size (size of the food)
d_iuom_unit_of_measure_code := iuom_unit_of_measure_code
If a record exists on i_foods with if_food_id_equiv = d_if_food_id
d_other_foods_use_as_equiv := 'Y'

v_cost := 0 -- cost of the d_if_food_id from the Vendor's food price survey (v_prices)

If d_other_foods_use_as_equiv = 'Y' (value is generic, like "cheese")

For each food that has if_food_id_equiv equal to d_if_food_id, (like "cheddar")

t_items = trunc(d_unit_size / unit_size)-- finds how many items can be combined without exceeding the d_unit_size.

Compute t_cost := v_prices.max_price for the food * t_items

If t_cost > v_cost
v_cost := t_cost

Else (the food is an actual food like "Kix")

v_cost = v_prices.max_price * t_items

End if

Update temp_ven_price_survey set maximum_price = maximum_price +
(V_cost * quantity)
where ven_id = d_ven_id and fi_type = d_ifit_fi_type_code

Delete records on the temporary table Temp_Ven_Price_Survey where the maximum_price is zero/null.

2. Rank by maximum FI price from Price Survey and generate/store FI redemption information for the Vendor/FI Type combination:

For the unique peer group and FI type combinations from Temp_Ven_Price_Survey, set local variables:

d_pee_peer_group := peer group
d_ifit_fi_type_code := FI type

d_rank := 1

Get the Vendor ID's for the peer group/FI type combination from Temp_Ven_Price_Survey, in descending order of the vendors' maximum_price for the FI type. Set local variable:

d_ven_d = vendor id

From the Vendor's redeemed FI's (idis_disposition_code = '4') for the FI Type and Peer Group, set local variables:

d_sum := sum(redemption_amt)
d_count := count(ifi.serial_number)
d_average := sum(redemption_amt)/count(ifi.serial_number)

Update temp_ven_price_survey set rank = d_rank,
redemption_volume = d_sum, fi_count = d_count,
avg_redemption_value = d_average

d_rank := d_rank + 1

3. Rank by average redeemed FI price for the Vendor

d_rank := 1

Get the Vendor ID's for the peer group/FI type combination from Temp_Ven_Price_Survey, in descending order of the vendors' avg_redemption_value for the FI type

Update temp_ven_price_survey set redemption_rank = d_rank

d_rank := d_rank + 1

4. Insert/Update v_fi_prices (Vendor's FI cost from Price Survey) and v_surveys (Vendor's Price Survey) tables

For all records in the temporary table Temp_Ven_Price_Survey, set local variables:

d_ven_id := Ven_id
d_ifit_fi_type_code := fi_type

Find the date of the most recent Price Survey for the Vendor: max(survey_date)

If no price survey (v_surveys) record was found for the Vendor, Insert into v_surveys set
survey_date = EOD run date,
submitted = 'Y'

If no FI Price Survey (v_fi_prices) record was found for the Vendor/FI, insert into v_fi_prices
set fi_maximum_price = Temp_Ven_Price_Survey.maximum_price,
redemption_rank = Temp_Ven_Price_Survey.redemption_rank,
rank_by_max = Temp_Ven_Price_Survey.rank

else
update v_fi_prices
set fi_maximum_price = Temp_Ven_Price_Survey.maximum_price,
redemption_rank = Temp_Ven_Price_Survey.redemption_rank,
rank_by_max = Temp_Ven_Price_Survey.rank

(Redemption_volume, fi_count, and avg_redemption_value are not stored in v_fi_prices.)

Delete all v_fi_prices records where (date_created is not the current date and date_modified is not the current date.

Drop Temp_Ven_Price_Survey

1.27.1.4 Outputs

v_fi_prices
v_surveys
Temp_Ven_Price_Survey

1.27.2 EC4_VEN_RISK_HISTORY.SQL

Overview

At the end of the fiscal year, a new V_RISK_LEVEL_HISTORY record must be created for the Applicant/Vendor, containing the new fiscal year and the current V_RISK_LEVELS.RISK_LEVEL_ID value.

Input

v_risk_level_histories

Selection

all v_risk_level_histories records for the current fiscal year

Processing

If the date of the end-of-day is the last day of the fiscal year (September 30, or the most recent business day)

find all v_risk_level_histories records with current fiscal year in v_risk_level_histories.fy_of_risk,

if a record does not exist for the current fiscal year plus 1,

duplicate the record, setting v_risk_level_histories.fy_of_risk to the current fiscal year, plus 1.

Outputs

v_risk_level_histories

1.28 Gather new/updated data to be sent to the agencies

1.28.1 EC4_TRNC.SQL

Overview

This script initializes the following tables:

- E_EOD_CONTROLS
- A_EOD_CONTROLS
- A_STATE_CLIENTS

1.28.1.1 Inputs

N/A

1.28.1.2 Selection

N/A

1.28.1.3 Processing

TRUNCATE TABLE E_EOD_CONTROLS
TRUNCATE TABLE A_EOD_CONTROLS
TRUNCATE TABLE A_STATE_CLIENTS

1.28.1.4 Outputs

EC4_TRNC.LOG

1.28.2 EC4_TAB_DOWN.SQL

Overview

This script runs for each Local Agency.,

The primary purpose of this script is to propagate down to Local Agencies changes made to selected tables at the State level. Temporary tables are utilized to facilitate data recovery efforts should they be needed.

For each temporary table, the old data in the table is truncated and new rows and values are inserted where either the date modified or the date created column of the "source" table is after the last successful run date for End Of Day Processing. For some temporary tables, additional criteria is employed.

1.28.2.1 Inputs

N/A

1.28.2.2 Selection

For all tables in this script, inserts are based on the date_created or date_modified being after the last successful run date of EOD. Some tables contain other conditions listed below.

For A_I_FOOD INSTRUMENTS:
If the Process Date is not null, and the Food Instrument is not associated with a Compliance Buy. The date modified is after the last successful run date of EOD Processing, and the clinic is in the local agency for which the script is being run

For A_I_FI_REJECT_REASONS:
If the Date Modified or Date Created is after the last successful run date of EOD Processing and the Food Instrument was issued by a clinic in the Local Agency for which the script is being run

For A_F_CASELOADS:
If the Date Modified or Date Created is after the last successful run date of EOD Processing and the Clinic is in the Local Agency for which the script is being run

For A_F_CASE_ASSIGNMENTS:
If the Date Modified or Date Created is after the last successful run date of EOD Processing and the Clinic is in the Local Agency for which the script is being run

For A_DEL_STATEMENTS:
If the Date Modified or Date Created is after the last successful run date of EOD Processing and target_proc_date is null and the originating agency is Central

For all other tables:
If the Date Modified or Date Created is after the last successful run date of EOD Processing

1.28.2.3 Processing

Truncate old data from temporary table.
Copy rows/columns into temporary table based on above selection criteria.

1.28.2.4 Outputs

1. A log file is generated for each script/Local Agency, i.e. 01_EC4_TAB.LOG, 02_EC4_TAB.LOG, etc. An example of this log file can be found in Appendix A.

2. The following temporary tables are created:
   - A_EOD_CONTROLS
   - A_A_ACTIVITIES
   - A_AAS_CLASS_CATEGORIES
   - A_AAS_ITEMS
   - A_A_APPT_ITEMS
   - A_A_ATTEND_STATUSES
   - A_A_OFFICE_CLOSED
   - A_A_SERVICES
   - A_I_AGE_RANGES
   - A_C_ANSWERS
   - A_C_ANSWER_TYPES
   - A_C_BLOODWORK_TYPES
   - A_C_BREAST_PUMP_REASONS
   - A_C_BREAST_PUMP_TYPES
   - A_C_CATEGORIES
   - A_C_CAT_BLOOD_FACTORS
   - A_C_CAT_GOALS
   - A_C_CAT_NUTR_EDS
   - A_C_CAT_REFERRALS
   - A_C_COMMUNICATION_TYPES
   - A_C_CP_MESSAGES
   - A_C_DIAGNOSES
   - A_C_DIETARY_REQUIREMENTS
   - A_C_DIET_NUTRIENT_TYPES
   - A_C_DISABILITIES
   - A_C_EDUCATION_LEVELS
   - A_C_ELEVATIONS
   - A_C_ETHNIC_GROUPS
   - A_C_GOALS
   - A_C_HC_PAYEES
   - A_C_HH_QUESTIONS
   - A_C_HH_RESPONSES
   - A_C_INCOME_INTERVALS

A_C_INCOME_LEVELS
A_F_POVERTY_BASES
A_C_INCOME_SOURCES
A_C_INCOME_VERIFICATIONS
A_C_INFANT_STATUSES
A_C_LANGUAGES
A_C_MARITAL_STATUSES
A_C_NCHS_CLASSIFICATIONS
A_C_NCHS_TYPES
A_C_NCHS_DATA
A_C_NO_CONTACT_REASONS
A_C_NUTR_ED_MATERIALS
A_C_NUTR_ED_TOPICS
A_C_PICKUP_INTERVALS
A_C_PILOT_QUESTIONS
A_C_PILOT_STUDIES
A_C_PRIORITIES
A_C_PROOF_ADDRESSES
A_C_PROOF_IDENTITIES
A_C_PS_QUESTIONS
A_C_RACES
A_C_REASONS_BF_ENDED
A_C_RESOLUTIONS
A_C_RESOLVED_CLIENTS
A_C_RISK_FACTORS
A_C_RISK_FACTOR_DIAGNOSES
A_C_RISK_FACTOR_TYPES
A_C_RF_GOALS
A_C_RF_NUTR_EDS
A_C_RF_REFERRALS
A_C_RF_SERVICES
A_C_HH_RESPONSE_RFS
A_C_BMI_DATA
A_C_BMI_ANTHROPOMETRIC
A_C_BMI_WEIGHT_GAIN
A_C_SCHEDULE_DAYS
A_C_SMOKING_CHANGES
A_C_SMOKING_STAGES
A_C_SOURCES_HEALTH_CARE
A_C_SYMPTOMS
A_C_TERM_REASONS
A_C_TOPICS
A_C_VOTER_REGISTRATIONS
A_C_CAT_BLOODWORKS
A_C_DESIREABLE_WEIGHTS
PROMPT EA1 inserting :A_C_IMMUNIZATIONS_NOT_ASSESSED
A_F_CONTROLS
A_F_CASELOAD_TYPES
A_F_FUND_SOURCES
A_F_POVERTY_LEVELS
A_F_WAIT_LIST_RESPONSES

A_I_BANK_DISPOSITIONS
A_I_CATEGORY_GROUPS
A_I_PACKAGES
A_I_PRODUCTS
A_I_REJECT_REASONS
A_I_CONTAINERS
A_I_DISPOSITIONS
A_I_UNITS_OF_MEASURE
A_I_VOID_REASONS
A_I_FOOD_GROUPS
A_I_FOODS
A_I_FOOD_DISTRIBUTIONS
A_I_MAXIMUM_FOODS
A_I_FOOD_PACKAGES
A_I_CATEGORY_GROUP_PKGS
A_I_FOOD_PACKAGE_FI_TYPES
A_I_FOOD_PACKAGE_FOODS
A_I_PACKAGE_DISTRIBUTIONS
A_I_WS_FOOD_GROUPS
A_I_FOOD_INSTRUMENT_TYPES
A_I_FOOD_INSTRUMENT_FOODS
A_C_FOOD_PKG_RISK_FACTORS
A_O_OUTREACH_ORG_TYPES
A_O_OUTREACH_COMM_TYPES
--A_O_OUTREACH_ORGANIZATIONS
--A_O_OUTREACH_ORG_PHONES
--A_O_OUTREACH_COMMS
A_O_PROGRAMS
A_O_PROGRAM_DATES
A_O_TITLE_CATEGORIES
A_O_STAFF_TITLES
A_S_CITIES
A_S_CONTACT_METHODS
A_S_CONTACT_TITLES
A_S_COUNTIES
A_S_PHONE_TYPES
A_S_STATES
A_S_ZIPS
A_S_GEO_LOCATIONS
A_V_ACTIVITY_TYPES
A_V_APPLICATION_MILESTONES
A_V_APPLICATION_TYPES
A_V_BANK_BRANCHES
A_V_CASE_STATUSES
A_V_COLLECTION_TYPES
A_V_COMMUNICATION_TYPES
A_V_COMPLAINT_SOURCE_TYPES
A_V_COMPLAINT_STATUSES
A_V_COMPLAINT_SUBJECTS
A_V_COMPLIANCE_CASE_DESIGNATNS
A_V_COMPLIANCE_CASE_TYPES

A_V_DELIVERY_TYPES
A_V_DENIAL_REASONS
A_V_DISQUAL_REASONS
A_V_FINDING_CODES
A_V_FSP_REGIONAL_OFFICES
A_V_FSP_VIOLATION_CODES
A_V_LEGAL_FIRMS
A_V_LEGAL_REPRESENTATIVES
A_V_MILESTONE_TYPES
A_V_OWNER_TYPES
A_V_PEER_GROUPS
A_V_RISK_LEVELS
A_V_SANCTION_TYPES
A_V_STATUSES
A_V_SUSPENSION_REASONS
A_V_VIOLATION_ACTION
A_V_WHOLESALERS
A_V_WIC_CODES
A_V_OWNERS
A_V_VENDORS
--A_O_STAFF_MEMBERS
A_O_ORGANIZATIONAL_UNITS
A_O_ANNUAL_WIC_COST_SUMMARIES
A_F_ANNUAL_FACTORS
A_F_BUDGETS
A_F_CASE_ASSIGNMENTS
A_F_CASELOADS
A_F_CASELOAD_RESTRICTIONS
A_F_MFR_CONTACTS
A_F_MANUFACTURERS
A_I_FI_INVENTORIES
A_I_STOCK_INVENTORIES
A_I_FOOD_INSTRUMENTS
A_I_FI_REJECT_REASONS
S_PRINTER_ADDRESSES
A_DEL_STATEMENTS

1.28.3 EC4_AGCY_BEING_PROCESSED.SQL

Overview

This script will indicate the agency for which data is updated on the State Database in the End of Day Process.

1.28.3.1 Inputs

Agency ID

1.28.3.2 Selection

N/A

1.28.3.3 Processing

Update env_variables
Set env_value = decode (agcy_id, ’01, ‘2’, ‘02’, ‘6’, ‘03’, ‘17’, ‘04’, ‘27’, ‘05’, ‘33’, ‘06’, ‘37’, ‘07’, ‘41’, ‘08’, ‘58’, ‘09’, ‘65’, ‘10’, ‘73’, ‘11’, ‘93’, ‘13’, ‘107’, ‘14’, ‘119’, ‘15’, ‘297’, ‘17’, ‘130’, ‘25’, ‘27’, ‘133’, ‘28’, ‘141’, ‘29’, ‘151’, ‘30’, ‘155’, ‘32’, 318)
Where code= ‘AGCY_BEING_PROCESSED’;

1.28.3.4 OUTPUTS

N/A

1.29 Prepare archival retrieval data

1.29.1 EC1_PRG_ARCHV.SQL

Overview

Purge archived client records marked with an archive flag of 'Y'. The deletes have been ordered from the lowest level (Level 6) child up to the parent level (Level 1). Purge the archived vendor records marked with an archive flag of 'Y'. The deletes have been ordered from the smallest level child up to the parent (v_vendors).

1.29.1.1 Inputs

s_client_archives sc
s_vendor_archives s
c_clients

1.29.1.2 Selection

Marks client records and vendor records that are to be deleted:
WHERE S.ARCHIVE_FLAG = 'Y'

1.29.1.3 Processing

Any client record marked with an archive flag of 'Y' is deleted from the following tables where the client ID is equal to the S_CLIENT_ARCHIVES.CLIENT_ID unless otherwise specified

LEVEL 6:
No deletions made at this level.

LEVEL 5:
A_APPT_TOPIC_MATERIALS

LEVEL 4:
A_APPT_TOPICS
A_APPT_APPT_ITEMS
C_CLIENT_SVC_NE_MATERIALS
AAS_APPTS_CLIENTS
F_WAIT_LIST_CONTACTS
C_DIET_NUTRIENTS
C_INCOMES where the Economic Member Code
WHERE ceum_economic_member_code in

```
            (SELECT economic_member_code
        FROM   c_economic_unit_members
        WHERE  cih_income_history_id in
          (SELECT income_history_id
            FROM   c_income_histories
            WHERE  cc_client_id = clid));
```

LEVEL 3:
A_APPOINTMENTS
C_CLIENT_NE_TOPICS
C_WOMAN_MEDICALS
C_BLOODWORK_DATA
C_CERT_TERM_REASONS
C_B_N_HEALTHS
C_FOOD_PACKAGE_PRESCRIPTIONS
F_WAIT_LISTS
C_INFANT_CHILD_MEDICALS
C_DIETARY_ASSESSMENTS
C_HEALTH_RISK_FACTORS
C_I_C_HEALTHS
C_ECONOMIC_UNIT_MEMBERS

```
        WHERE cih_income_history_id in
        (SELECT income_history_id
         FROM   c_income_histories
         WHERE  cc_client_id = clid);
```

C_PEER_RBFENDS
C_CONTACTS
C_P_COUNS_NOTES
I_BF_VCHR_ITEMS

```
        (SELECT voucher_no from I_BF_PROMO_VCHRS
         where cc_client_id = clid);
```

LEVEL 2:

I_BF_PROMO_VCHRS
C_WOMAN_MEDICALS
C_W_HEALTHS
C_INFANT_CHILD_MEDICALS
C_I_C_HEALTHS
C_BLOODWORK_DATA
C_PREV_FAMILIES
C_PREV_NAMES
C_ALLERGY_FOODS
C_MORE_ETHNIC_GROUPS
C_CLIENT_SERVICES
C_CERTIFICATIONS
C_TRANSFER_HISTORIES
C_IMMUNIZATIONS
C_CLIENT_REFERRALS
C_CLIENT_COMMUNICATIONS
C_RESOLUTIONS

C_CLIENT_NOTES
C_CLIENT_PROGS
C_CLIENT_GROUPS
C_INCOME_HISTORIES
C_INFANT_DATA
C_CLIENT_GOALS
C_CERT_PEER_COUNSELS

LEVEL 1:
C_CLIENTS
C_INCOMES
C_ECONOMIC_UNIT_MEMBERS
C_INCOME_HISTORIES

The S_CLIENT_ARCHIVES table is then updated by setting the ARCHIVE_FLAG = 'N'.

Insert record in R_ARCHIVED_CLIENTS.

Any vendor record marked with an archive flag of 'Y' is deleted from the following tables where the vendor ID is equal to the S_VENDOR_ARCHIVES.ID

LEVEL 6:
F_PURCHASE_ORDER_LINE_ITEMS
```
    where (fpoi_fpo_po_number, fpoi_invoice_date, fpoi_invoice_number ) in
    (select fpo_po_number, invoice_date, invoice_number
      from f_purchase_order_invoices
      where ifi_serial_number in
      (select serial_number
        from i_food_instruments
        where ven_id = vid));
```

LEVEL 5:
F_PURCHASE_ORDER_INVOICES
```
    where ifi_serial_number in
    (select serial_number
       from i_food_instruments
       where ven_id = vid);
```
I_FI_REJECT_REASONS
```
    where ifi_serial_number in
    (select serial_number
       from i_food_instruments
       where ven_id = vid);
```
I_FI_FORMULAS
```
    where ifi_serial_number in
    (select serial_number
       from i_food_instruments
       where ven_id = vid);
```

LEVEL 4:
V__CMP_PAYMENT_SCHEDULES

LEVEL 3:
V_FI_PRICES
V_SANCTIONS
V_ACTIVITY_FINDINGS
V_COMPLIANCE_CASE_CLIENTS
    where vcc_case_id in
    (select case_id
      from v_compliance_cases
      where ven_id = vid);
V_VEN_WS_FOOD_GROUPS
V_DENIAL_REASONS
V_AUTHORIZATION_MILESTONES
V_PRICES
V_REQUIRED_FOLLOWUPS
V_SUSPENSIONS
V_DISQUALIFICATIONS
V_CIVIL_MONEY_PENALTIES
V_APPEAL_STEPS

LEVEL 2:
I_FOOD_INSTRUCTIONS
V_APPEALS
V_MON_ACTIVITIES
V_VENDOR_ACCOUNTS
V_SURVEYS
V_VENDOR_PHONES
V_EDUCATIONS
V_HOURS_OF_OPERATIONS
V_COMPLIANCE_CASES
V_VENDOR_FSP_VIOLATIONS
V_RISK_LEVEL_HISTORIES
V_COLLECTIONS
V_VENDOR_WHOLESALERS
V_LA_CLINICS
V_AUTHORIZATIONS
V_COMPLAINTS
V_COMMUNICATIONS
V_OVERALL_RANKS
V_FELONIES
V_MULTI_WIC
V_WIC_VIOLATIONS
V_PS_RESPONSES

LEVEL 1:
v_vendors

The S_VENDOR_ARCHIVES table is then updated by setting the ARCHIVE_FLAG to ‘N’

1.29.1.4 Outputs

F_PURCHASE_ORDER_LINE_ITEMS
A_APPT_TOPIC_MATERIALS
A_APPT_TOPICS
A_APPT_APPTITEMS
C_CLIENT_SVC_NE_MATERIALS
C_B_N_HH_REASONS_BFENDS
F_WAIT_LIST_CONTACTS
C_DIET_NUTRIENTS
C_I_C_HH_REASONS_BFENDS
C_INCOMES
A_APPOINTMENTS
C_CLIENT_NE_TOPICS
C_WOMAN_MEDICALS
C_BLOODWORK_DATA
C_CERT_TERM_REASONS
C_B_N_HEALTHS
C_FOOD_PACKAGE_PRESCRIPTIONS
F_WAIT_LISTS
C_INFANT_CHILD_MEDICALS
C_DIETARY_ASSESSMENTS
C_HEALTH_RISK_FACTORS
C_I_C_HEALTHS
C_ECONOMIC_UNIT_MEMBERS
C_PEER_RBFENDS
C_CONTACTS
C_P_COUNS_NOTES
I_BF_VCHR_ITEMS
I_BF_PROMO_VCHRS
C_CLIENT_SERVICES
C_CERTIFICATIONS
C_TRANSFER_HISTORIES
C_IMMUNIZATIONS
C_CLIENT_REFERRALS
C_CLIENT_COMMUNICATIONS
C_RESOLUTIONS
C_CLIENT_NOTES
C_CLIENT_PROGS
C_CLIENT_GROUPS
C_INCOME_HISTORIES
C_INFANT_DATA
C_CLIENT_GOALS
C_CERT_PEER_COUNSELS
F_PURCHASE_ORDER_INVOICES
I_FI_REJECT_REASONS
I_FI_FORMULAS
S_CLIENT_ARCHIVES
S_VENDOR_ARCHIVES
V_PAYMENT_SCHEDULES
I_FOOD_INSTRUMENTS
V_SANCTIONS

V_ACTIVITY_FINDINGS
I_BF_VCHR_ITEMS
V_FI_PRICES
V_COMPLIANCE_CASE_CLIENTS
V_APPEALS
V_MON_ACTIVITIES
I_BF_PROMO_VCHRS
V_VEN_WS_FOOD_GROUPS
V_DENIAL_REASONS
V_AUTHORIZATION_MILESTONES
V_PRICES
V_SURVEYS
V_ACCOUNTS
V_VENDOR_PHONES
V_EDUCATIONS
V_HOURS_OF_OPERATIONS
V_COMPLIANCE_CASES
I_BF_PROMO_PAYMENTS
V_VENDOR_HEALTHS
V_VENDOR_WHOLESALERS
V_LA_CLINICS
V_AUTHORIZATIONS
V_COMPLAINTS
V_COMMUNICATIONS
V_OVERALL_RANKS

1.29.2 EC4_RETRIEVE.SQL

Overview

Run the following scripts (in system Administration):

* EC4_TRUNC_ARCH_CLI.SQL (reference System Administration DTSDSection 6 - 1.1.4.1)
* EC4_INS_RETR_CLI.SQL
* EC4_TRUNC_ARCH_VEN.SQL (reference System Administration DTSDSection 6 - 1.1.4.1)
* EC4_INS_RETR_VEN.SQL

1.29.2.1 Inputs

N/A

1.29.2.2 Selection

N/A

1.29.2.3 Processing

The following scripts are run:
EC4_TRUNC_ARCH_CLI.SQL
EC4_INS_RETR_CLI.SQL
EC4_TRUNC_ARCH_VEN.SQL
EC4_INS_RETR_VEN.SQL

1.29.2.4 Outputs

%AGCY%EC4_RETRIEVE.LOG

1.29.3 EC4_INS_RETR_CLI.SQL

Overview

Performs insert from Client tables into all temporary tables for retrieval of archived information. These tables are temporarily storing this information for transfer back into the AIM Certification tables. The temporary tables have a prefix of 'R' which represents that the tables are storing archived information that is being retrieved

r_c_clients
r_c_bloodwork_data
r_c_w_hh_reasons_bfends
r_c_certifications
r_c_cert_peer_counsels
r_c_cert_term_reasons
r_c_client_communications
r_c_client_goals
r_c_client_ne_topics
r_c_client_notes
r_c_client_progs
r_c_client_referrals
r_c_client_services
r_c_client_svc_ne_materials
r_c_contacts
r_c_dietary_assessments
r_c_diet_nutrients
r_c_economic_unit_members
r_c_family_economic_units
r_c_family_phones
r_c_fam_referrals
r_c_feu_communications
r_c_food_pack_scripts
r_c_health_risk_factors
r_c_immunizations
r_c_incomes
r_c_income_histories
r_c_infant_child_medicals
r_c_infant_data
r_c_i_c_healths
r_c_i_c_hh_reasons_bfends
r_c_peer_rbfends
r_c_p_couns_notes
r_c_w_healths
r_c_resolutions
r_c_transfer_histories
r_c_woman_medicals
r_s_client_archives

1.29.3.1 Inputs

C_CLIENTS CC
S_CLIENT_ARCHIVES SCA

1.29.3.2 Selection

```
WHERE  sca.client_id = cc.client_id
        AND    SCA.ARCHIVE_FLAG = 'R'
        AND    (TRUNC(sca.date_modified) >
             (SELECT good_proc_date FROM e_eod_controls) )
        ORDER BY cc.client_id;
```

This where clause selects where S_CLIENT_ARCHIVES.CLIENT_ID equals the C_CLIENTS.CLIENT_ID and the ARCHIVE_FLAG indicates that information has been archived and is being retrieved (‘R’) and the date modified is greater than the last successful procedure date. The ARCHIVE_FLAG is set equal to ‘R’ by first setting the flag to ‘Y’ which indicates that a record is to be archived. After the information has been deleted and archived by the system the ARCHIVE_FLAG is set to ‘N’ which is then set to ‘R’ when a retrieval request is made.

1.29.3.3 Processing

IF search_clients%FOUND THEN

INSERT INTO (list of output tables, each field in table)

1.29.3.4 Outputs

r_c_clients
r_c_bloodwork_data
r_c_w_healths
r_c_b_n_hh_reasons_bfends
r_c_certifications
r_c_cert_peer_counsels
r_c_cert_term_reasons
r_c_client_communications
r_c_client_goals
r_c_client_ne_topics
r_c_client_notes
r_c_client_progs
r_c_client_referrals
r_c_client_services

r_c_client_svc_ne_materials
r_c_contacts
r_c_dietary_assessments
r_c_diet_nutrients
r_c_economic_unit_members
r_c_family_economic_units
r_c_family_phones
r_c_fam_referrals
r_c_feu_communications
r_c_food_pack_scripts
r_c_health_risk_factors
r_c_immunizations
r_c_incomes
r_c_income_histories
r_c_infant_child_medicals
r_c_infant_data
r_c_i_c_healths
r_c_i_c_hh_reasons_bfends
r_c_peer_rbfends
r_c_p_couns_notes
r_c_w_healths
r_c_resolutions
r_c_transfer_histories
r_c_woman_medicals
r_s_client_archives

1.29.4 EC4_INS_RETR_VEN.SQL

Overview

Performs insert from Vendor tables into all temporary tables.

r_v_vendors
r_v_accounts
r_v_activity_findings
r_v_appeals
r_v_authorizations
r_v_auth_milestones
r_v_communications
r_v_complaints
r_v_compliance_cases
r_v_denial_reasons
r_v_educations
r_v_fi_prices
r_v_hours_of_operations
r_v_la_clinics
r_v_mon_activities
r_v_overall_ranks

r_v_owners
r_v_owner_phones
r_v_payment_schedules
r_v_prices
r_v_sanctions
r_v_surveys
r_v_vendor_fsps
r_v_vendor_healths
r_v_vendor_phones
r_v_vendor_wholesalers
r_v_ven_ws_food_groups
r_s_vendor_archives

1.29.4.1 Inputs

V_VENDORS VV
S_VENDOR_ARCHIVES SVV

1.29.4.2 Selection

```
WHERE  vv.id = svv.id
          AND    SVV.ARCHIVE_FLAG = 'R'
          AND    (TRUNC(svv.date_modified) >
               (SELECT good_proc_date FROM e_eod_controls) )
      ORDER BY vv.id
```

This where clause selects where V_VENDORS.ID equals the S_VENDOR_ARCHIVES.ID and the ARCHIVE_FLAG indicates that information has been archived and is being retrieved (‘R’) and the date modified is greater than the last successful procedure date. The ARCHIVE_FLAG is set equal to ‘R’ by first setting the flag to ‘Y’ which indicates that a record is to be archived. After the information has been deleted and archived by the system the ARCHIVE_FLAG is set to ‘N’ which is then set to ‘R’ when a retrieval request is made.

1.29.4.3 Processing

IF search_vendors%FOUND THEN

INSERT INTO (list of output tables)

1.29.4.4 Outputs

r_v_vendors
r_v_accounts

r_v_activity_findings
r_v_appeals
r_v_authorizations
r_v_auth_milestones
r_v_communications
r_v_complaints
r_v_compliance_cases
r_v_denial_reasons
r_v_educations
r_v_fi_prices
r_v_hours_of_operations
r_v_la_clinics
r_v_mon_activities
r_v_overall_ranks
r_v_owners
r_v_owner_phones
r_v_payment_schedules
r_v_prices
r_v_sanctions
r_v_surveys
r_v_vendor_fsps
r_v_vendor_healths
r_v_vendor_phones
r_v_vendor_wholesalers
r_v_ven_ws_food_groups
r_s_vendor_archives

1.30 Print out status logs

1.30.1 EC5.BAT

Overview

This batch file initiates the printing of Central log files.

Processing

Called by EC1.BAT.

The script first reads in the EC4.INI file to initialize and set up temporary environmental variables.

It then calls the EC_PRINT.BAT batch file which combines Central processing logs for output.

1.30.2 EC_PRINT.BAT

Overview

This batch file produces master log files by agency for the following log files:

* Truncate
* Import
* Delete
* Insert
* Update
* End of day import
* End of day insert
* End of day retrieve
* Export

1.30.2.1 Inputs

N/A

1.30.2.2 Selection

Called by EC5.BAT

1.30.2.3 Processing

The system sets the environmental variables for the particular Local Agency reading them from the EC4.INI file and places them in a temporary file to be used for retrieving scripts.

Prints out combined log files for the following files:

1. Combines the EC.LOG with the EC_HIS.LOG by writing it to the end of the EC_HIS.LOG file.

2. Writes the EC1_SQL.LOG, EC2_SQL.LOG, EC4_BANK_SQL.LOG, EC4_SQL.LOG, EC4_TRNC.LOG and %_EC2B_SQL.LOG files to the end of the CTRL_SQL.LOG (new file each day) and SQL.LOG (history file of CTRL_SQL.LOG).

3. Then creates the files AGCY_%.LOG (new file each day) and AGCY_%_HIS.LOG (history file of AGCY_%.LOG) that log the truncate, import, delete - insert - update - delete, EOD import, EOD insert, EOD retrieve, and export information. The following logs for each Local Agency are written to the end of the AGCY_%.LOG and AGCY_%_HIS.LOG (the wildcard represents the Local Agencies ID):

The truncate section: %_TRUNC.LOG
The import section: %_IMP.LOG
The delete - insert - update - delete section: %_EC2_AGENCY.LOG
The EOD import section: %EOD_IMP.LOG
The EOD insert section: %_EC4_TAB.LOG
The EOD retrieve section: %_EC4_RETRIEVE.LOG
The export section: %EXP.LOG

The final combined files are EC_HIS.LOG, AGCY_CTRL.LOG, CTRL_SQL.LOG, SQL.LOG, AGCY_%_HIS.LOG and AGCY_%.LOG. Examples of these log files can be found in Appendix A.

1.30.2.4 Outputs

Agcy_%_his.log. See Appendix A for example of the layout.

1.31 Consolidate information from central

1.31.1 EA3.BAT

Overview

This batch file runs at the Local Agencies after Central processing has been completed.

Processing

The system sets the environmental variables for the particular Local Agency reading them from the EA1.INI file and places them in a temporary file.
This script retrieves the return DMP file from central by running EA3_FTP_RUN.BAT, then searches for a completion flag (%.don) from the particular Agency to begin loading the Agencies data from the DMP file.

The agency is then imported by starting the following SQL scripts:
- EA_TRGOFF.SQL which disables all triggers and constraints.
- EA3_TRNC.SQL which initializes the tables.

After importing is completed EA3_PRE.SQL is started which updates organizational units staff member id to null if it's not in the current agency.

The following SQL scripts are then initiated to delete, insert and update the Agency:
- EA3_DELTRIG.SQL
- EA3_IN_TAB.SQL
- EA3_TAB_UP.SQL
- EA3_SYNC_LA_ASSIGNMENT.SQL
- EA3_TRGON.SQL

Once the Agency has been updated EA3.SQL and EA3_RETRIEVE.SQL are executed.
EA3.SQL runs the following SQL scripts:
- EA3_TERM_TR_CLIENTS.SQL
- EA3_PROC_FI.SQL
- EA3_PRG_ARCHV.SQL

EA3_RETRIEVE.SQL runs the following scripts:
- EA3_INS_RETR_CLI.SQL
- EA3_INS_RETR_VEN.SQL

The script then executes the batch file EA_PRINT.BAT.

The script finishs by executing EA3_DBA.SQL to monitor DBA activities.

1.31.2 EA_TRGOFF.SQL

Overview

Sets database triggers off, so that table contents can be modified without restrictions.

1.31.2.1 Inputs

N/A

1.31.2.2 Selection

N/A

1.31.2.3 Processing

Alter the following tables with 'DISABLE ALL TRIGGERS'

F_ANNUAL_FACTORS
I_FOOD_INSTRUMENT_TYPES
C_FOOD_PACKAGE_PRESCRIPTIONS
C_CLIENT_SERVICES

Alter the following tables with 'DISABLE CONSTRAINT'
I_FOOD_INSTRMENT_TYPES with constraint IFIT_IFIT_FK
I_FOOD_PACKAGE_FI_TYPES with constraint IFPFT_IFPFT_FK
F_MANUFACTURERS with constraint MAN_FMC_FK
F_MFR_CONTACTS with constraint FMC_MAN_FK
C_RESOLUTIONS with constraint CR1_CC_FK

1.31.2.4 Outputs

N/A

1.31.3 EA3_TRNC.SQL

Overview

This script empties out the a_* and r_* tables before they are filled with data sent from the Central Processor to the Agencies.

1.31.3.1 Inputs

The following is a list of tables that are truncated by this script:

A_STATE_CLIENTS
A_C_STATE_CLIENTS
A_F_POVERTY_BASES
A_EOD_CONTROLS
A_A_ACTIVITIES
A_A_APPT_ITEMS
A_AAS_CLASS_CATEGORIES
A_AAS_ITEMS
A_A_ATTEND_STATUSES
A_A_OFFICE_CLOSED
A_A_SERVICES
A_I_AGE_RANGES
A_C_ANSWERS
A_C_ANSWER_TYPES
A_C_BLOODWORK_TYPES
A_C_BREAST_PUMP_REASONS
A_C_BREAST_PUMP_TYPES
A_C_CATEGORIES
A_C_CAT_BLOOD_FACTORS
A_C_CAT_GOALS
A_C_CAT_NUTR_EDS
A_C_CAT_REFERRALS
A_C_COMMUNICATION_TYPES
A_C_CP_MESSAGES
A_C_DIAGNOSES
A_C_DIETARY_REQUIREMENTS
A_C_DIET_NUTRIENT_TYPES
A_C_DISABILITIES
A_C_EDUCATION_LEVELS
A_C_ELEVATIONS
A_C_ETHNIC_GROUPS
A_C_GOALS
A_C_HC_PAYEES
A_C_HH_QUESTIONS
A_C_HH_RESPONSES
A_C_INCOME_INTERVALS
A_C_INCOME_LEVELS
A_C_INCOME_SOURCES
A_C_INCOME_VERIFICATIONS
A_C_INFANT_STATUSES
A_C_LANGUAGES
A_C_MARITAL_STATUSES
A_C_NCHS_CLASSIFICATIONS
A_C_NCHS_TYPES
A_C_NCHS_DATA
A_C_NO_CONTACT_REASONS

A_C_NUTR_ED_MATERIALS
A_C_NUTR_ED_TOPICS
A_C_PICKUP_INTERVALS
A_C_PILOT_QUESTIONS
A_C_PILOT_STUDIES
A_C_PRIORITIES
A_C_PROOF_ADDRESSES
A_C_PROOF_IDENTITIES
A_C_PS_QUESTIONS
A_C_RACES
A_C_REASONS_BF_ENDED
A_C_RISK_FACTORS
A_C_RISK_FACTOR_DIAGNOSES
A_C_RISK_FACTOR_TYPES
A_C_RF_GOALS
A_C_RF_NUTR_EDS
A_C_RF_REFERRALS
A_C_RF_SERVICES
A_C_HH_RESPONSE_RFS
A_C_BMI_DATA
A_C_BMI_ANTHROPOMETRIC
A_C_BMI_WEIGHT_GAIN
A_C_SCHEDULE_DAYS
A_C_SMOKING_CHANGES
A_C_SMOKING_STAGES
A_C_SOURCES_HEALTH_CARE
A_C_SYMPTOMS
A_C_TERM_REASONS
A_C_TOPICS
A_C_VOTER_REGISTRATIONS
A_C_CAT_BLOODWORKS
A_C_DESIREABLE_WEIGHTS
A_C_IMMUNIZATIONS_NOT_ASSESSED
A_F_CONTROLS
A_F_CASELOAD_TYPES
A_F_FUND_SOURCES
A_F_POVERTY_LEVELS
A_F_WAIT_LIST_RESPONSES
A_I_BANK_DISPOSITIONS
A_I_CATEGORY_GROUPS
A_I_PACKAGES
A_I_PRODUCTS
A_I_REJECT_REASONS
A_I_CONTAINERS
A_I_DISPOSITIONS
A_I_UNITS_OF_MEASURE
A_I_VOID_REASONS
A_I_FOOD_GROUPS
A_I_FOODS
A_I_FOOD_DISTRIBUTIONS
A_I_MAXIMUM_FOODS

A_I_FOOD_PACKAGES
A_I_CATEGORY_GROUP_PKGS
A_I_FOOD_PACKAGE_FI_TYPES
A_I_FOOD_PACKAGE_FOODS
A_I_PACKAGE_DISTRIBUTIONS
A_I_WS_FOOD_GROUPS
A_I_FOOD_INSTRUMENT_TYPES
A_I_FOOD_INSTRUMENT_FOODS
A_C_FOOD_PKG_RISK_FACTORS
A_O_OUTREACH_ORG_TYPES
A_O_OUTREACH_COMM_TYPES
A_O_OUTREACH_ORGANIZATIONS
A_O_OUTREACH_ORG_PHONES
A_O_OUTREACH_COMMS
A_O_PROGRAMS
A_O_PROGRAM_DATES
A_O_TITLE_CATEGORIES
A_O_STAFF_TITLES
A_S_CITIES
A_S_CONTACT_METHODS
A_S_CONTACT_TITLES
A_S_COUNTIES
A_S_PHONE_TYPES
A_S_STATES
A_S_ZIPS
A_V_ACTIVITY_TYPES
A_V_APPLICATION_MILESTONES
A_V_APPLICATION_TYPES
A_V_BANK_BRANCHES
A_V_CASE_STATUSES
A_V_COLLECTION_TYPES
A_V_COMMUNICATION_TYPES
A_V_COMPLAINT_SOURCE_TYPES
A_V_COMPLAINT_STATUSES
A_V_COMPLAINT_SUBJECTS
A_V_COMPLIANCE_CASE_DESIGNATNS
A_V_COMPLIANCE_CASE_TYPES
A_V_DELIVERY_TYPES
A_V_DENIAL_REASONS
A_V_DISQUAL_REASONS
A_V_FINDING_CODES
A_V_FSP_REGIONAL_OFFICES
A_V_FSP_VIOLATION_CODES
A_V_LEGAL_FIRMS
A_V_LEGAL_REPRESENTATIVES
A_V_MILESTONE_TYPES
A_V_OWNER_TYPES
A_V_PEER_GROUPS
A_V_RISK_LEVELS
A_V_SANCTION_TYPES
A_V_STATUSES

A_V_SUSPENSION_REASONS
A_V_VIOLATION_ACTION
A_V_WHOLESALERS
A_V_WIC_CODES
A_V_OWNERS
A_V_VENDORS
A_O_STAFF_MEMBERS
A_O_ORGANIZATIONAL_UNITS
A_O_ANNUAL_WIC_COST_SUMMARIES
A_F_ANNUAL_FACTORS
A_F_BUDGETS
A_F_CASE_ASSIGNMENTS
A_F_CASELOADS
A_F_CASELOAD_RESTRICTIONS
A_F_MFR_CONTACTS
A_F_MANUFACTURERS
A_I_FI_INVENTORIES
A_I_STOCK_INVENTORIES
A_I_FOOD_INSTRUMENTS
A_I_FI_REJECT_REASONS
A_C_RESOLUTIONS
A_C_RESOLVED_CLIENTS
A_S_GEO_LOCATIONS;
A_S_PRINTER_ADDRESSES;
A_DEL_STATEMENTS
R_S_CLIENT_ARCHIVES
PROMPT table containing clients who have been retrieved from the archives
R_ARCHIVED_CLIENTS
PROMPT table containing clients who have been archived at central

1.31.3.2 Selection

All records are deleted from the tables listed.

1.31.3.3 Processing

Performs an Oracle TRUNCATE on each of the tables, removing all records in the tables.

1.31.3.4 Outputs

Same as Inputs (Section 1.32.3.1.)

1.31.4 EA3_PRE.SQL

Overview

This script updates organizational units staff member ID to null if it's not in the current agency.

1.31.4.1 Input

A_O_ORGANIZATIONAL_UNITS

1.31.4.2 Selection

Updates staff member's ID to NULL if it does not belong to the current agency:

For the a_o_organizational_units table:

WHERE OU_SEQ_ID is not the Server's local agency sequence number
AND SEQ_ID is not the Server's local agency sequence number

The Server's local agency is defined as: SELECT ENV('AGENCY_SEQ') FROM DUAL

1.31.4.3 Processing

UPDATE A_O_ORGANIZATIONAL_UNITS SET SM_STAFF_MEMBER_ID = NULL

1.31.4.4 Output

A_O_ORGANIZATIONAL_UNITS

1.31.5 EA3_DELTRIG.SQL

Overview

This script deletes data from the local agency database that had been deleted from the central database since the last end of day run.

1.31.5.1 Inputs

A_DEL_STATEMENTS

1.31.5.2 Selection

for a_del_statements:

```
WHERE   target_proc_date IS NULL
ORDER BY origin_agcy, del_seq
```

1.31.5.3 Processing

Loop through all selected records on table a_del_statements:

write out the a_del_statements.del_statement to a log

set local variables:

```
cursor_var := DBMS_SQL.OPEN_CURSOR;
delete_statement := a_del_statements.del_statement;
```

send the command to SQL:

```
DBMS_SQL.PARSE(cursor_var, DELETE_STATEMENT, dbms_sql.v7);
```

capture the return value and close the SQL:

```
ret_value := DBMS_SQL.EXECUTE(cursor_var);
DBMS_SQL.CLOSE_CURSOR(cursor_var);
```

End loop;

update the a_del_statements, setting the target_proc_date to the current date where the target_proc_date is null

1.31.5.4 Outputs

a_del_statements

message to print the delete statements processed:

EA3_DELTRIG.LOG

1.31.6 EA3_IN_TAB.SQL

Overview

This script inserts records into AIM tables from End of Day admin tables.

A_ACTIVITIES
AAS_CLASS_CATEGORIES
AAS_ITEMS
A_APPT_ITEMS
A_ATTEND_STATUSES
A_OFFICE_CLOSED
A_SERVICES
I_AGE_RANGES
C_ANSWERS
C_ANSWER_TYPES
C_BLOODWORK_TYPES
C_BREAST_PUMP_REASONS
C_BREAST_PUMP_TYPES
C_CATEGORIES
C_CAT_BLOOD_FACTORS
C_CAT_GOALS
C_CAT_NUTR_EDS
C_CAT_REFERRALS
C_COMMUNICATION_TYPES
C_CP_MESSAGES
C_DIAGNOSES
C_DIETARY_REQUIREMENTS
C_DIET_NUTRIENT_TYPES
C_DISABILITIES
C_EDUCATION_LEVELS
C_ELEVATIONS
C_ETHNIC_GROUPS
C_GOALS
C_HC_PAYEES
C_HH_QUESTIONS
C_HH_RESPONSES
C_INCOME_INTERVALS
C_INCOME_LEVELS
F_POVERTY_BASES
C_INCOME_SOURCES
C_INCOME_VERIFICATIONS
C_INFANT_STATUSES
C_LANGUAGES
C_MARITAL_STATUSES
C_NCHS_CLASSIFICATIONS
C_NCHS_TYPES
C_NCHS_DATA

C_NO_CONTACT_REASONS
C_NUTR_ED_MATERIALS
C_NUTR_ED_TOPICS
C_PICKUP_INTERVALS
C_PILOT_QUESTIONS
C_PILOT_STUDIES
C_PRIORITIES
C_PROOF_ADDRESSES
C_PROOF_IDENTITIES
C_PS_QUESTIONS
C_RACES
C_REASONS_BF_ENDED
C_RISK_FACTORS
C_RISK_FACTOR_DIAGNOSES
C_RISK_FACTOR_TYPES
C_RF_GOALS
C_RF_NUTR_EDS
C_RF_REFERRALS
C_RF_SERVICES
C_HH_RESPONSE_RFS
C_BMI_DATA
C_BMI_ANTHROPOMETRIC
C_BMI_WEIGHT_GAIN
C_SCHEDULE_DAYS
C_SMOKING_CHANGES
C_SMOKING_STAGES
C_SOURCES_HEALTH_CARE
C_SYMPTOMS
C_TERM_REASONS
C_TOPICS
C_VOTER_REGISTRATIONS
C_CAT_BLOODWORKS
C_DESIREABLE_WEIGHTS
C_IMMUNIZATIONS_NOT_ASSESSED
F_CONTROLS
F_CASELOAD_TYPES
F_FUND_SOURCES
F_POVERTY_LEVELS
F_WAIT_LIST_RESPONSES
I_BANK_DISPOSITIONS
I_CATEGORY_GROUPS
I_PACKAGES
I_PRODUCTS
I_REJECT_REASONS
I_CONTAINERS
I_DISPOSITIONS
I_UNITS_OF_MEASURE
I_VOID_REASONS
I_FOOD_GROUPS
I_FOODS
I_FOOD_DISTRIBUTIONS

I_MAXIMUM_FOODS
I_FOOD_PACKAGES
I_CATEGORY_GROUP_PKGS
I_FOOD_PACKAGE_FOODS
I_PACKAGE_DISTRIBUTIONS
I_WS_FOOD_GROUPS
I_FOOD_INSTRUMENT_TYPES
I_FOOD_PACKAGE_FI_TYPES
I_FOOD_INSTRUMENT_FOODS
C_FOOD_PKG_RISK_FACTORS
O_OUTREACH_ORG_TYPES
O_OUTREACH_COMM_TYPES
O_PROGRAMS
O_PROGRAM_DATES
O_TITLE_CATEGORIES
O_STAFF_TITLES
S_CITIES
S_CONTACT_METHODS
S_CONTACT_TITLES
S_COUNTIES
S_PHONE_TYPES
S_STATES
S_ZIPS
S_GEO_LOCATIONS
V_ACTIVITY_TYPES
V_APPLICATION_MILESTONES
V_APPLICATION_TYPES
V_BANK_BRANCHES
V_CASE_STATUSES
V_COLLECTION_TYPES
V_COMMUNICATION_TYPES
V_COMPLAINT_SOURCE_TYPES
V_COMPLAINT_STATUSES
V_COMPLAINT_SUBJECTS
V_COMPLIANCE_CASE_DESIGNATIONS
V_COMPLIANCE_CASE_TYPES
V_DELIVERY_TYPES
V_DENIAL_REASONS
V_DISQUAL_REASONS
V_FINDING_CODES
V_FSP_REGIONAL_OFFICES
V_FSP_VIOLATION_CODES
V_LEGAL_FIRMS
V_LEGAL_REPRESENTATIVES
V_MILESTONE_TYPES
V_OWNER_TYPES
V_PEER_GROUPS
V_RISK_LEVELS
V_SANCTION_TYPES
V_STATUSES
V_SUSPENSION_REASONS

V_VIOLATION_ACTION
V_WHOLESALERS
V_WIC_CODES
V_OWNERS
V_VENDORS
O_STAFF_MEMBERS
PROMPT EA3 updating :A_O_ORGANIZATIONAL_UNITS.SM_STAFF_MEMBER_ID TO NULL
O_ORGANIZATIONAL_UNITS
O_ANNUAL_WIC_COST_SUMMARIES
F_ANNUAL_FACTORS
F_BUDGETS
F_CASE_ASSIGNMENTS
F_CASELOADS
F_CASELOAD_RESTRICTIONS
F_MFR_CONTACTS
F_MANUFACTURERS
I_FI_INVENTORIES
I_STOCK_INVENTORIES
C_RESOLVED_CLIENTS
C_RESOLUTIONS
I_FI_REJECT_REASONS
S_PRINTER_ADDRESSES
DEL_STATEMENTS

1.31.6.1 Inputs

Records are inserted from the following tables:

A_EOD_CONTROLS
A_A_ACTIVITIES
A_AAS_CLASS_CATEGORIES
A_AAS_ITEMS
A_A_APPT_ITEMS
A_A_ATTEND_STATUSES
A_A_OFFICE_CLOSED
A_A_SERVICES
A_I_AGE_RANGES
A_C_ANSWERS
A_C_ANSWER_TYPES
A_C_BLOODWORK_TYPES
A_C_BREAST_PUMP_REASONS
A_C_BREAST_PUMP_TYPES
A_C_CATEGORIES
A_C_CAT_BLOOD_FACTORS
A_C_CAT_GOALS
A_C_CAT_NUTR_EDS
A_C_CAT_REFERRALS
A_C_COMMUNICATION_TYPES

A_C_CP_MESSAGES
A_C_DIAGNOSES
A_C_DIETARY_REQUIREMENTS
A_C_DIET_NUTRIENT_TYPES
A_C_DISABILITIES
A_C_EDUCATION_LEVELS
A_C_ELEVATIONS
A_C_ETHNIC_GROUPS
A_C_GOALS
A_C_HC_PAYEES
A_C_HH_QUESTIONS
A_C_HH_RESPONSES
A_C_INCOME_INTERVALS
A_C_INCOME_LEVELS
A_F_POVERTY_BASES
A_C_INCOME_SOURCES
A_C_INCOME_VERIFICATIONS
A_C_INFANT_STATUSES
A_C_LANGUAGES
A_C_MARITAL_STATUSES
A_C_NCHS_CLASSIFICATIONS
A_C_NCHS_TYPES
A_C_NCHS_DATA
A_C_NO_CONTACT_REASONS
A_C_NUTR_ED_MATERIALS
A_C_NUTR_ED_TOPICS
A_C_PICKUP_INTERVALS
A_C_PILOT_QUESTIONS
A_C_PILOT_STUDIES
A_C_PRIORITIES
A_C_PROOF_ADDRESSES
A_C_PROOF_IDENTITIES
A_C_PS_QUESTIONS
A_C_RACES
A_C_REASONS_BF_ENDED
A_C_RESOLUTIONS
A_C_RESOLVED_CLIENTS
A_C_RISK_FACTORS
A_C_RISK_FACTOR_DIAGNOSES
A_C_RISK_FACTOR_TYPES
A_C_RF_GOALS
A_C_RF_NUTR_EDS
A_C_RF_REFERRALS
A_C_RF_SERVICES
A_C_HH_RESPONSE_RFS
A_C_BMI_DATA
A_C_BMI_ANTHROPOMETRIC
A_C_BMI_WEIGHT_GAIN
A_C_SCHEDULE_DAYS
A_C_SMOKING_CHANGES
A_C_SMOKING_STAGES

A_C_SOURCES_HEALTH_CARE
A_C_SYMPTOMS
A_C_TERM_REASONS
A_C_TOPICS
A_C_VOTER_REGISTRATIONS
A_C_CAT_BLOODWORKS
A_C_DESIREABLE_WEIGHTS
PROMPT EA1 inserting :A_C_IMMUNIZATIONS_NOT_ASSESSED
A_F_CONTROLS
A_F_CASELOAD_TYPES
A_F_FUND_SOURCES
A_F_POVERTY_LEVELS
A_F_WAIT_LIST_RESPONSES
A_I_BANK_DISPOSITIONS
A_I_CATEGORY_GROUPS
A_I_PACKAGES
A_I_PRODUCTS
A_I_REJECT_REASONS
A_I_CONTAINERS
A_I_DISPOSITIONS
A_I_UNITS_OF_MEASURE
A_I_VOID_REASONS
A_I_FOOD_GROUPS
A_I_FOODS
A_I_FOOD_DISTRIBUTIONS
A_I_MAXIMUM_FOODS
A_I_FOOD_PACKAGES
A_I_CATEGORY_GROUP_PKGS
A_I_FOOD_PACKAGE_FI_TYPES
A_I_FOOD_PACKAGE_FOODS
A_I_PACKAGE_DISTRIBUTIONS
A_I_WS_FOOD_GROUPS
A_I_FOOD_INSTRUMENT_TYPES
A_I_FOOD_INSTRUMENT_FOODS
A_C_FOOD_PKG_RISK_FACTORS
A_O_OUTREACH_ORG_TYPES
A_O_OUTREACH_COMM_TYPES
--A_O_OUTREACH_ORGANIZATIONS
--A_O_OUTREACH_ORG_PHONES
--A_O_OUTREACH_COMMS
A_O_PROGRAMS
A_O_PROGRAM_DATES
A_O_TITLE_CATEGORIES
A_O_STAFF_TITLES
A_S_CITIES
A_S_CONTACT_METHODS
A_S_CONTACT_TITLES
A_S_COUNTIES
A_S_PHONE_TYPES
A_S_STATES
A_S_ZIPS

A_S_GEO_LOCATIONS
A_V_ACTIVITY_TYPES
A_V_APPLICATION_MILESTONES
A_V_APPLICATION_TYPES
A_V_BANK_BRANCHES
A_V_CASE_STATUSES
A_V_COLLECTION_TYPES
A_V_COMMUNICATION_TYPES
A_V_COMPLAINT_SOURCE_TYPES
A_V_COMPLAINT_STATUSES
A_V_COMPLAINT_SUBJECTS
A_V_COMPLIANCE_CASE_DESIGNATNS
A_V_COMPLIANCE_CASE_TYPES
A_V_DELIVERY_TYPES
A_V_DENIAL_REASONS
A_V_DISQUAL_REASONS
A_V_FINDING_CODES
A_V_FSP_REGIONAL_OFFICES
A_V_FSP_VIOLATION_CODES
A_V_LEGAL_FIRMS
A_V_LEGAL_REPRESENTATIVES
A_V_MILESTONE_TYPES
A_V_OWNER_TYPES
A_V_PEER_GROUPS
A_V_RISK_LEVELS
A_V_SANCTION_TYPES
A_V_STATUSES
A_V_SUSPENSION_REASONS
A_V_VIOLATION_ACTION
A_V_WHOLESALERS
A_V_WIC_CODES
A_V_OWNERS
A_V_VENDORS
--A_O_STAFF_MEMBERS
A_O_ORGANIZATIONAL_UNITS
A_O_ANNUAL_WIC_COST_SUMMARIES
A_F_ANNUAL_FACTORS
A_F_BUDGETS
A_F_CASE_ASSIGNMENTS
A_F_CASELOADS
A_F_CASELOAD_RESTRICTIONS
A_F_MFR_CONTACTS
A_F_MANUFACTURERS
A_I_FI_INVENTORIES
A_I_STOCK_INVENTORIES
A_I_FOOD_INSTRUMENTS
A_I_FI_REJECT_REASONS
S_PRINTER_ADDRESSES
A_DEL_STATEMENTS

1.31.6.2 Selection

WHERE TRUNC (date_created) > (date of last successful EOD))

1.31.6.3 Processing

Inserts rows into the agency server's tables by selecting the records from the corresponding a_* table:

```
Insert into aim.table
        (...columns...)
        (select ...columns...
                from eodadm.a_table
                where the date_created is after the last successful end of day run)

where table is the name of a table, such as a_attend_statuses
 and columns is the matching names of the columns in the tables.
```

For example:

```
Insert into aim.a_attend_statuses
        (attend_status_code, description, date_created, created_by, date_modified, modified_by,
                note)
        (Select attend_status_code, description, date_created, created_by, date_modified,
                modified_by, note
        from eodadm.a_a_attend_statuses
        where trunc(date_created) > (select good_proc_date from eodadm.a_eod_controls) )
```

1.31.6.4 Outputs

A_ACTIVITIES
AAS_CLASS_CATEGORIES
AAS_ITEMS
A_APPT_ITEMS
A_ATTEND_STATUSES
A_OFFICE_CLOSED
A_SERVICES
I_AGE_RANGES
C_ANSWERS
C_ANSWER_TYPES
C_BLOODWORK_TYPES
C_BREAST_PUMP_REASONS
C_BREAST_PUMP_TYPES
C_CATEGORIES
C_CAT_BLOOD_FACTORS

C_CAT_GOALS
C_CAT_NUTR_EDS
C_CAT_REFERRALS
C_COMMUNICATION_TYPES
C_CP_MESSAGES
C_DIAGNOSES
C_DIETARY_REQUIREMENTS
C_DIET_NUTRIENT_TYPES
C_DISABILITIES
C_EDUCATION_LEVELS
C_ELEVATIONS
C_ETHNIC_GROUPS
C_GOALS
C_HC_PAYEES
C_HH_QUESTIONS
C_HH_RESPONSES
C_INCOME_INTERVALS
C_INCOME_LEVELS
F_POVERTY_BASES
C_INCOME_SOURCES
C_INCOME_VERIFICATIONS
C_INFANT_STATUSES
C_LANGUAGES
C_MARITAL_STATUSES
C_NCHS_CLASSIFICATIONS
C_NCHS_TYPES
C_NCHS_DATA
C_NO_CONTACT_REASONS
C_NUTR_ED_MATERIALS
C_NUTR_ED_TOPICS
C_PICKUP_INTERVALS
C_PILOT_QUESTIONS
C_PILOT_STUDIES
C_PRIORITIES
C_PROOF_ADDRESSES
C_PROOF_IDENTITIES
C_PS_QUESTIONS
C_RACES
C_REASONS_BF_ENDED
C_RISK_FACTORS
C_RISK_FACTOR_DIAGNOSES
C_RISK_FACTOR_TYPES
C_RF_GOALS
C_RF_NUTR_EDS
C_RF_REFERRALS
C_RF_SERVICES
C_HH_RESPONSE_RFS
C_BMI_DATA
C_BMI_ANTHROPOMETRIC
C_BMI_WEIGHT_GAIN
C_SCHEDULE_DAYS

C_SMOKING_CHANGES
C_SMOKING_STAGES
C_SOURCES_HEALTH_CARE
C_SYMPTOMS
C_TERM_REASONS
C_TOPICS
C_VOTER_REGISTRATIONS
C_CAT_BLOODWORKS
C_DESIREABLE_WEIGHTS
C_IMMUNIZATIONS_NOT_ASSESSED
F_CONTROLS
F_CASELOAD_TYPES
F_FUND_SOURCES
F_POVERTY_LEVELS
F_WAIT_LIST_RESPONSES
I_BANK_DISPOSITIONS
I_CATEGORY_GROUPS
I_PACKAGES
I_PRODUCTS
I_REJECT_REASONS
I_CONTAINERS
I_DISPOSITIONS
I_UNITS_OF_MEASURE
I_VOID_REASONS
I_FOOD_GROUPS
I_FOODS
I_FOOD_DISTRIBUTIONS
I_MAXIMUM_FOODS
I_FOOD_PACKAGES
I_CATEGORY_GROUP_PKGS
I_FOOD_PACKAGE_FOODS
I_PACKAGE_DISTRIBUTIONS
I_WS_FOOD_GROUPS
I_FOOD_INSTRUMENT_TYPES
I_FOOD_PACKAGE_FI_TYPES
I_FOOD_INSTRUMENT_FOODS
C_FOOD_PKG_RISK_FACTORS
O_OUTREACH_ORG_TYPES
O_OUTREACH_COMM_TYPES
O_PROGRAMS
O_PROGRAM_DATES
O_TITLE_CATEGORIES
O_STAFF_TITLES
S_CITIES
S_CONTACT_METHODS
S_CONTACT_TITLES
S_COUNTIES
S_PHONE_TYPES
S_STATES
S_ZIPS
S_GEO_LOCATIONS

V_ACTIVITY_TYPES
V_APPLICATION_MILESTONES
V_APPLICATION_TYPES
V_BANK_BRANCHES
V_CASE_STATUSES
V_COLLECTION_TYPES
V_COMMUNICATION_TYPES
V_COMPLAINT_SOURCE_TYPES
V_COMPLAINT_STATUSES
V_COMPLAINT_SUBJECTS
V_COMPLIANCE_CASE_DESIGNATIONS
V_COMPLIANCE_CASE_TYPES
V_DELIVERY_TYPES
V_DENIAL_REASONS
V_DISQUAL_REASONS
V_FINDING_CODES
V_FSP_REGIONAL_OFFICES
V_FSP_VIOLATION_CODES
V_LEGAL_FIRMS
V_LEGAL_REPRESENTATIVES
V_MILESTONE_TYPES
V_OWNER_TYPES
V_PEER_GROUPS
V_RISK_LEVELS
V_SANCTION_TYPES
V_STATUSES
V_SUSPENSION_REASONS
V_VIOLATION_ACTION
V_WHOLESALERS
V_WIC_CODES
V_OWNERS
V_VENDORS
O_STAFF_MEMBERS
PROMPT EA3 updating :A_O_ORGANIZATIONAL_UNITS.SM_STAFF_MEMBER_ID TO NULL
O_ORGANIZATIONAL_UNITS
O_ANNUAL_WIC_COST_SUMMARIES
F_ANNUAL_FACTORS
F_BUDGETS
F_CASE_ASSIGNMENTS
F_CASELOADS
F_CASELOAD_RESTRICTIONS
F_MFR_CONTACTS
F_MANUFACTURERS
I_FI_INVENTORIES
I_STOCK_INVENTORIES
C_RESOLVED_CLIENTS
C_RESOLUTIONS
I_FI_REJECT_REASONS
S_PRINTER_ADDRESSES
DEL_STATEMENTS

1.31.7 EA3_TAB_UP.SQL

Overview

This script performs updates to the following tables:

A_ACTIVITIES
AAS_CLASS_CATEGORIES
AAS_ITEMS
A_APPT_ITEMS
A_ATTEND_STATUSES
A_OFFICE_CLOSED
A_SERVICES
I_AGE_RANGES
C_ANSWERS
C_ANSWER_TYPES
C_BLOODWORK_TYPES
C_BREAST_PUMP_REASONS
C_BREAST_PUMP_TYPES
C_CATEGORIES
C_CAT_BLOOD_FACTORS
C_CAT_GOALS
C_CAT_NUTR_EDS
C_CAT_REFERRALS
C_COMMUNICATION_TYPES
C_CP_MESSAGES
C_DIAGNOSES
C_DIETARY_REQUIREMENTS
C_DIET_NUTRIENT_TYPES
C_DISABILITIES
C_EDUCATION_LEVELS
C_ELEVATIONS
C_ETHNIC_GROUPS
C_GOALS
C_HC_PAYEES
C_HH_QUESTIONS
C_HH_RESPONSES
C_INCOME_INTERVALS
C_INCOME_LEVELS
F_POVERTY_BASES
C_INCOME_SOURCES
C_INCOME_VERIFICATIONS
C_INFANT_STATUSES
C_LANGUAGES
C_MARITAL_STATUSES
C_NCHS_CLASSIFICATIONS
C_NCHS_TYPES
C_NCHS_DATA

C_NO_CONTACT_REASONS
C_NUTR_ED_MATERIALS
C_NUTR_ED_TOPICS
C_PICKUP_INTERVALS
C_PILOT_QUESTIONS
C_PILOT_STUDIES
C_PRIORITIES
C_PROOF_ADDRESSES
C_PROOF_IDENTITIES
C_PS_QUESTIONS
C_RACES
C_REASONS_BF_ENDED
C_RISK_FACTORS
C_RISK_FACTOR_DIAGNOSES
C_RISK_FACTOR_TYPES
C_RF_GOALS
C_RF_NUTR_EDS
C_RF_REFERRALS
C_RF_SERVICES
C_HH_RESPONSE_RFS
C_BMI_DATA
C_BMI_ANTHROPOMETRIC
C_BMI_WEIGHT_GAIN
C_SCHEDULE_DAYS
C_SMOKING_CHANGES
C_SMOKING_STAGES
C_SOURCES_HEALTH_CARE
C_SYMPTOMS
C_TERM_REASONS
C_TOPICS
C_VOTER_REGISTRATIONS
C_CAT_BLOODWORKS
C_DESIREABLE_WEIGHTS
C_IMMUNIZATIONS_NOT_ASSESSED
F_CONTROLS
F_CASELOAD_TYPES
F_FUND_SOURCES
F_POVERTY_LEVELS
F_WAIT_LIST_RESPONSES
I_BANK_DISPOSITIONS
I_CATEGORY_GROUPS
I_PACKAGES
I_PRODUCTS
I_REJECT_REASONS
I_CONTAINERS
I_DISPOSITIONS
I_UNITS_OF_MEASURE
I_VOID_REASONS
I_FOOD_GROUPS
I_FOODS
I_FOOD_DISTRIBUTIONS

I_MAXIMUM_FOODS
I_FOOD_PACKAGES
I_CATEGORY_GROUP_PKGS
I_FOOD_PACKAGE_FOODS
I_PACKAGE_DISTRIBUTIONS
I_WS_FOOD_GROUPS
I_FOOD_INSTRUMENT_TYPES
I_FOOD_PACKAGE_FI_TYPES
I_FOOD_INSTRUMENT_FOODS
C_FOOD_PKG_RISK_FACTORS
O_OUTREACH_ORG_TYPES
O_OUTREACH_COMM_TYPES
O_PROGRAMS
O_PROGRAM_DATES
O_TITLE_CATEGORIES
O_STAFF_TITLES
S_CITIES
S_CONTACT_METHODS
S_CONTACT_TITLES
S_COUNTIES
S_PHONE_TYPES
S_STATES
S_ZIPS
S_GEO_LOCATIONS
V_ACTIVITY_TYPES
V_APPLICATION_MILESTONES
V_APPLICATION_TYPES
V_BANK_BRANCHES
V_CASE_STATUSES
V_COLLECTION_TYPES
V_COMMUNICATION_TYPES
V_COMPLAINT_SOURCE_TYPES
V_COMPLAINT_STATUSES
V_COMPLAINT_SUBJECTS
V_COMPLIANCE_CASE_DESIGNATIONS
V_COMPLIANCE_CASE_TYPES
V_DELIVERY_TYPES
V_DENIAL_REASONS
V_DISQUAL_REASONS
V_FINDING_CODES
V_FSP_REGIONAL_OFFICES
V_FSP_VIOLATION_CODES
V_LEGAL_FIRMS
V_LEGAL_REPRESENTATIVES
V_MILESTONE_TYPES
V_OWNER_TYPES
V_PEER_GROUPS
V_RISK_LEVELS
V_SANCTION_TYPES
V_STATUSES
V_SUSPENSION_REASONS

V_VIOLATION_ACTION
V_WHOLESALERS
V_WIC_CODES
V_OWNERS
V_VENDORS
O_STAFF_MEMBERS
PROMPT EA3 updating :A_O_ORGANIZATIONAL_UNITS.SM_STAFF_MEMBER_ID TO NULL
O_ORGANIZATIONAL_UNITS
O_ANNUAL_WIC_COST_SUMMARIES
F_ANNUAL_FACTORS
F_BUDGETS
F_CASE_ASSIGNMENTS
F_CASELOADS
F_CASELOAD_RESTRICTIONS
F_MFR_CONTACTS
F_MANUFACTURERS
I_FI_INVENTORIES
I_STOCK_INVENTORIES
C_RESOLVED_CLIENTS
C_RESOLUTIONS
I_FI_REJECT_REASONS
S_PRINTER_ADDRESSES
DEL_STATEMENTS

1.31.7.1 Inputs

The following temporary tables are used to make the updates:

A_EOD_CONTROLS
A_A_ACTIVITIES
A_AAS_CLASS_CATEGORIES
A_AAS_ITEMS
A_A_APPT_ITEMS
A_A_ATTEND_STATUSES
A_A_OFFICE_CLOSED
A_A_SERVICES
A_I_AGE_RANGES
A_C_ANSWERS
A_C_ANSWER_TYPES
A_C_BLOODWORK_TYPES
A_C_BREAST_PUMP_REASONS
A_C_BREAST_PUMP_TYPES
A_C_CATEGORIES
A_C_CAT_BLOOD_FACTORS
A_C_CAT_GOALS
A_C_CAT_NUTR_EDS
A_C_CAT_REFERRALS

A_C_COMMUNICATION_TYPES
A_C_CP_MESSAGES
A_C_DIAGNOSES
A_C_DIETARY_REQUIREMENTS
A_C_DIET_NUTRIENT_TYPES
A_C_DISABILITIES
A_C_EDUCATION_LEVELS
A_C_ELEVATIONS
A_C_ETHNIC_GROUPS
A_C_GOALS
A_C_HC_PAYEES
A_C_HH_QUESTIONS
A_C_HH_RESPONSES
A_C_INCOME_INTERVALS
A_C_INCOME_LEVELS
A_F_POVERTY_BASES
A_C_INCOME_SOURCES
A_C_INCOME_VERIFICATIONS
A_C_INFANT_STATUSES
A_C_LANGUAGES
A_C_MARITAL_STATUSES
A_C_NCHS_CLASSIFICATIONS
A_C_NCHS_TYPES
A_C_NCHS_DATA
A_C_NO_CONTACT_REASONS
A_C_NUTR_ED_MATERIALS
A_C_NUTR_ED_TOPICS
A_C_PICKUP_INTERVALS
A_C_PILOT_QUESTIONS
A_C_PILOT_STUDIES
A_C_PRIORITIES
A_C_PROOF_ADDRESSES
A_C_PROOF_IDENTITIES
A_C_PS_QUESTIONS
A_C_RACES
A_C_REASONS_BF_ENDED
A_C_RESOLUTIONS
A_C_RESOLVED_CLIENTS
A_C_RISK_FACTORS
A_C_RISK_FACTOR_DIAGNOSES
A_C_RISK_FACTOR_TYPES
A_C_RF_GOALS
A_C_RF_NUTR_EDS
A_C_RF_REFERRALS
A_C_RF_SERVICES
A_C_HH_RESPONSE_RFS
A_C_BMI_DATA
A_C_BMI_ANTHROPOMETRIC
A_C_BMI_WEIGHT_GAIN
A_C_SCHEDULE_DAYS
A_C_SMOKING_CHANGES

A_C_SMOKING_STAGES
A_C_SOURCES_HEALTH_CARE
A_C_SYMPTOMS
A_C_TERM_REASONS
A_C_TOPICS
A_C_VOTER_REGISTRATIONS
A_C_CAT_BLOODWORKS
A_C_DESIREABLE_WEIGHTS
PROMPT EA1 inserting :A_C_IMMUNIZATIONS_NOT_ASSESSED
A_F_CONTROLS
A_F_CASELOAD_TYPES
A_F_FUND_SOURCES
A_F_POVERTY_LEVELS
A_F_WAIT_LIST_RESPONSES
A_I_BANK_DISPOSITIONS
A_I_CATEGORY_GROUPS
A_I_PACKAGES
A_I_PRODUCTS
A_I_REJECT_REASONS
A_I_CONTAINERS
A_I_DISPOSITIONS
A_I_UNITS_OF_MEASURE
A_I_VOID_REASONS
A_I_FOOD_GROUPS
A_I_FOODS
A_I_FOOD_DISTRIBUTIONS
A_I_MAXIMUM_FOODS
A_I_FOOD_PACKAGES
A_I_CATEGORY_GROUP_PKGS
A_I_FOOD_PACKAGE_FI_TYPES
A_I_FOOD_PACKAGE_FOODS
A_I_PACKAGE_DISTRIBUTIONS
A_I_WS_FOOD_GROUPS
A_I_FOOD_INSTRUMENT_TYPES
A_I_FOOD_INSTRUMENT_FOODS
A_C_FOOD_PKG_RISK_FACTORS
A_O_OUTREACH_ORG_TYPES
A_O_OUTREACH_COMM_TYPES
--A_O_OUTREACH_ORGANIZATIONS
--A_O_OUTREACH_ORG_PHONES
--A_O_OUTREACH_COMMS
A_O_PROGRAMS
A_O_PROGRAM_DATES
A_O_TITLE_CATEGORIES
A_O_STAFF_TITLES
A_S_CITIES
A_S_CONTACT_METHODS
A_S_CONTACT_TITLES
A_S_COUNTIES
A_S_PHONE_TYPES
A_S_STATES

A_S_ZIPS
A_S_GEO_LOCATIONS
A_V_ACTIVITY_TYPES
A_V_APPLICATION_MILESTONES
A_V_APPLICATION_TYPES
A_V_BANK_BRANCHES
A_V_CASE_STATUSES
A_V_COLLECTION_TYPES
A_V_COMMUNICATION_TYPES
A_V_COMPLAINT_SOURCE_TYPES
A_V_COMPLAINT_STATUSES
A_V_COMPLAINT_SUBJECTS
A_V_COMPLIANCE_CASE_DESIGNATNS
A_V_COMPLIANCE_CASE_TYPES
A_V_DELIVERY_TYPES
A_V_DENIAL_REASONS
A_V_DISQUAL_REASONS
A_V_FINDING_CODES
A_V_FSP_REGIONAL_OFFICES
A_V_FSP_VIOLATION_CODES
A_V_LEGAL_FIRMS
A_V_LEGAL_REPRESENTATIVES
A_V_MILESTONE_TYPES
A_V_OWNER_TYPES
A_V_PEER_GROUPS
A_V_RISK_LEVELS
A_V_SANCTION_TYPES
A_V_STATUSES
A_V_SUSPENSION_REASONS
A_V_VIOLATION_ACTION
A_V_WHOLESALERS
A_V_WIC_CODES
A_V_OWNERS
A_V_VENDORS
--A_O_STAFF_MEMBERS
A_O_ORGANIZATIONAL_UNITS
A_O_ANNUAL_WIC_COST_SUMMARIES
A_F_ANNUAL_FACTORS
A_F_BUDGETS
A_F_CASE_ASSIGNMENTS
A_F_CASELOADS
A_F_CASELOAD_RESTRICTIONS
A_F_MFR_CONTACTS
A_F_MANUFACTURERS
A_I_FI_INVENTORIES
A_I_STOCK_INVENTORIES
A_I_FOOD_INSTRUMENTS
A_I_FI_REJECT_REASONS
S_PRINTER_ADDRESSES
A_DEL_STATEMENTS

1.31.7.2 Selection

WHERE date_modified > (SELECT good_proc_date FROM EODADM.a_eod_controls))

1.31.7.3 Processing

Update existing records on the tables listed in the Inputs section (above). Selects records from the eodadm.a_* tables that have their date_modified after the last successful EOD run, matches them to a record on the agency tables, and updates the agency tables with the eodadm.a_* record contents:

```
update table
        set (...columns...) =
                (select ...columns...
                        from eodadm.a_table
                        where date_modified > (last EOD)
                        and table.pk = eodadm.a_table.pk )
        where (pk) in
                (select pk from eodadm.a_table where date_modified > last EOD)
```

where table is the name of a table, such as c_goals
columns is the matching names of the columns in the table
pk is the primary key to a table

For example:

```
Update c_goals
        set (description,date_created,created_by,date_modified,modified_by,note) = (Select
        description,date_created,created_by,date_modified,modified_by,note
                        from eodadm.a_c_goals
                        where date_modified > (last EOD)
                        and c_goals.goal_code = eodadm.a_c_goals.goal_code )
        where (c_goals.goal_code) in
                (select goal_code from eodadm.a_c_goals where trunc(date_modified) > (last
EOD))
```

1.31.7.4 Outputs

A_ACTIVITIES
AAS_CLASS_CATEGORIES
AAS_ITEMS
A_APPT_ITEMS
A_ATTEND_STATUSES
A_OFFICE_CLOSED
A_SERVICES

I_AGE_RANGES
C_ANSWERS
C_ANSWER_TYPES
C_BLOODWORK_TYPES
C_BREAST_PUMP_REASONS
C_BREAST_PUMP_TYPES
C_CATEGORIES
C_CAT_BLOOD_FACTORS
C_CAT_GOALS
C_CAT_NUTR_EDS
C_CAT_REFERRALS
C_COMMUNICATION_TYPES
C_CP_MESSAGES
C_DIAGNOSES
C_DIETARY_REQUIREMENTS
C_DIET_NUTRIENT_TYPES
C_DISABILITIES
C_EDUCATION_LEVELS
C_ELEVATIONS
C_ETHNIC_GROUPS
C_GOALS
C_HC_PAYEES
C_HH_QUESTIONS
C_HH_RESPONSES
C_INCOME_INTERVALS
C_INCOME_LEVELS
F_POVERTY_BASES
C_INCOME_SOURCES
C_INCOME_VERIFICATIONS
C_INFANT_STATUSES
C_LANGUAGES
C_MARITAL_STATUSES
C_NCHS_CLASSIFICATIONS
C_NCHS_TYPES
C_NCHS_DATA
C_NO_CONTACT_REASONS
C_NUTR_ED_MATERIALS
C_NUTR_ED_TOPICS
C_PICKUP_INTERVALS
C_PILOT_QUESTIONS
C_PILOT_STUDIES
C_PRIORITIES
C_PROOF_ADDRESSES
C_PROOF_IDENTITIES
C_PS_QUESTIONS
C_RACES
C_REASONS_BF_ENDED
C_RISK_FACTORS
C_RISK_FACTOR_DIAGNOSES
C_RISK_FACTOR_TYPES
C_RF_GOALS

C_RF_NUTR_EDS
C_RF_REFERRALS
C_RF_SERVICES
C_HH_RESPONSE_RFS
C_BMI_DATA
C_BMI_ANTHROPOMETRIC
C_BMI_WEIGHT_GAIN
C_SCHEDULE_DAYS
C_SMOKING_CHANGES
C_SMOKING_STAGES
C_SOURCES_HEALTH_CARE
C_SYMPTOMS
C_TERM_REASONS
C_TOPICS
C_VOTER_REGISTRATIONS
C_CAT_BLOODWORKS
C_DESIREABLE_WEIGHTS
C_IMMUNIZATIONS_NOT_ASSESSED
F_CONTROLS
F_CASELOAD_TYPES
F_FUND_SOURCES
F_POVERTY_LEVELS
F_WAIT_LIST_RESPONSES
I_BANK_DISPOSITIONS
I_CATEGORY_GROUPS
I_PACKAGES
I_PRODUCTS
I_REJECT_REASONS
I_CONTAINERS
I_DISPOSITIONS
I_UNITS_OF_MEASURE
I_VOID_REASONS
I_FOOD_GROUPS
I_FOODS
I_FOOD_DISTRIBUTIONS
I_MAXIMUM_FOODS
I_FOOD_PACKAGES
I_CATEGORY_GROUP_PKGS
I_FOOD_PACKAGE_FOODS
I_PACKAGE_DISTRIBUTIONS
I_WS_FOOD_GROUPS
I_FOOD_INSTRUMENT_TYPES
I_FOOD_PACKAGE_FI_TYPES
I_FOOD_INSTRUMENT_FOODS
C_FOOD_PKG_RISK_FACTORS
O_OUTREACH_ORG_TYPES
O_OUTREACH_COMM_TYPES
O_PROGRAMS
O_PROGRAM_DATES
O_TITLE_CATEGORIES
O_STAFF_TITLES

S_CITIES
S_CONTACT_METHODS
S_CONTACT_TITLES
S_COUNTIES
S_PHONE_TYPES
S_STATES
S_ZIPS
S_GEO_LOCATIONS
V_ACTIVITY_TYPES
V_APPLICATION_MILESTONES
V_APPLICATION_TYPES
V_BANK_BRANCHES
V_CASE_STATUSES
V_COLLECTION_TYPES
V_COMMUNICATION_TYPES
V_COMPLAINT_SOURCE_TYPES
V_COMPLAINT_STATUSES
V_COMPLAINT_SUBJECTS
V_COMPLIANCE_CASE_DESIGNATIONS
V_COMPLIANCE_CASE_TYPES
V_DELIVERY_TYPES
V_DENIAL_REASONS
V_DISQUAL_REASONS
V_FINDING_CODES
V_FSP_REGIONAL_OFFICES
V_FSP_VIOLATION_CODES
V_LEGAL_FIRMS
V_LEGAL_REPRESENTATIVES
V_MILESTONE_TYPES
V_OWNER_TYPES
V_PEER_GROUPS
V_RISK_LEVELS
V_SANCTION_TYPES
V_STATUSES
V_SUSPENSION_REASONS
V_VIOLATION_ACTION
V_WHOLESALERS
V_WIC_CODES
V_OWNERS
V_VENDORS
O_STAFF_MEMBERS
PROMPT EA3 updating :A_O_ORGANIZATIONAL_UNITS.SM_STAFF_MEMBER_ID TO NULL
O_ORGANIZATIONAL_UNITS
O_ANNUAL_WIC_COST_SUMMARIES
F_ANNUAL_FACTORS
F_BUDGETS
F_CASE_ASSIGNMENTS
F_CASELOADS
F_CASELOAD_RESTRICTIONS
F_MFR_CONTACTS
F_MANUFACTURERS

I_FI_INVENTORIES
I_STOCK_INVENTORIES
C_RESOLVED_CLIENTS
C_RESOLUTIONS
I_FI_REJECT_REASONS
S_PRINTER_ADDRESSES
DEL_STATEMENTS

1.31.8 EA_TRGON.SQL

Overview

Sets database triggers on, so that triggers and foreign keys are re-enabled.

1.31.8.1 Inputs

N/A

1.31.8.2 Selection

N/A

1.31.8.3 Processing

Alter the following tables with 'ENABLE ALL TRIGGERS'

F_ANNUAL_FACTORS
I_FOOD_INSTRUMENT_TYPES
C_FOOD_PACKAGE_PRESCRIPTIONS
C_CLIENT_SERVICES

Alter the following tables with 'ENABLE CONSTRAINT'

I_FOOD_INSTRMENT_TYPES with constraint IFIT_IFIT_FK
I_FOOD_PACKAGE_FI_TYPES with constraint IFPFT_IFPFT_FK
F_MANUFACTURERS with constraint MAN_FMC_FK
F_MFR_CONTACTS with constraint FMC_MAN_FK
C_RESOLUTIONS with constraint CR1_CC_FK

1.31.8.4 Outputs

N/A

1.32 Clinic serial number replenish

1.32.1 EC4_CL_SER_REP.SQL

Overview

During the end of day process the system automatically replenishes serial numbers for the clinics with the number of serial numbers at 10% or less of the number defined by the static factors, “small clinic FI’s”, “med clinic FI’s”, and “large clinic FI’s”.

1.32.1.1 Inputs

I_FI_INVENTORIES
F_CONTROLS
O_ORGANIZATIONAL_UNITS

1.32.1.2 Selection

MAX(I_FI_INVENTORIES.END_NO) for each clinic where
(I_FI_INVENTORIES.END_NO - I_FI_INVENTORIES.LAST_FI_NO_USED ≤ (.1) * F_CONTROLS.SMALL_CLINIC_FIS and O_ORGANIZATIONAL_UNITS.ORG_SIZE = ‘SMALL’) OR
(I_FI_INVENTORIES.END_NO - I_FI_INVENTORIES.LAST_FI_NO_USED ≤ (.1) * F_CONTROLS.MED_CLINIC_FIS and O_ORGANIZATIONAL_UNITS.ORG_SIZE = ‘MEDIUM’) OR
(I_FI_INVENTORIES.END_NO - I_FI_INVENTORIES.LAST_FI_NO_USED ≤ (.1) * F_CONTROLS.LARGE_CLINIC_FIS and O_ORGANIZATIONAL_UNITS.ORG_SIZE = ‘LARGE’)

1.32.1.3 Processing

For each I_FI_INVENTORIES record meeting the selection criteria
Loop

```
If O_ORGANIZATIONAL_UNITS.ORG_SIZE = ‘SMALL’
        number_of_fis  = F_CONTROLS.SMALL_CLINIC_FIS
Elsif O_ORGANIZATIONAL_UNITS.ORG_SIZE = ‘MEDIUM’
        number_of_fis  = F_CONTROLS.MED_CLINIC_FIS
Else
        number_of_fis  = F_CONTROLS.LARGE_CLINIC_FIS
End if
```

Insert into I_FI_INVENTORIES at the central database setting the columns as follows:

START_NO = MAX(I_FI_INVENTORIES.END_NO) + 1
END_NO = START_NO + number_of_fis
NO_ISSUED = number_of_fis
LAST_FI_NO_USED = START_NO - 1

Copy this I_FI_INVENTORIES record to the L/A database of the clinic

End Loop

1.32.1.4 Outputs

I_FI_INVENTORIES

1.33 Update caseload assignment information from the clinics

1.33.1 EA3_SYNC_LA_ASSIGNMENT.SQL

Overview

This script is used at local agencies for end of day processing to synchronize the caseload assignment data that has been manually re-allocated by the staff. This script needs to be executed before the end of day administrative tables are deleted, as it uses the end of day administrative f_case_assignment table to insert and/or update data into f_case_assignments at the local agency.

1.33.1.1 Inputs

f_case_assignments
a_f_case_assignments

1.33.1.2 Processing

For all F_CASE_ASSIGNMENTS records at the local agencies where the primary keys are equal to those in EODADM.A_F_CASE_ASSIGNMENTS (a temporary table holding Caseload Assignments that have changed at the state level)

Update F_CASE_ASSIGNMENTS

Set Assigned, Alloc Factor, Date Created, Created By, Date Modified, and Modified By to their equivalents in EODADM.A_F_CASE_ASSIGNMENTS.

For new case assignments insert new record in F_CASE_ASSIGNMENTS table.

1.33.1.3 Outputs

Log file of which assignments have changed. Log file is EA3_SYNC_LA_ASSIGNMENT.LOG

f_case_assignments

1.33.2 EA3.SQL

Overview

Runs SQL scripts for EA3.BAT.

1.33.2.1 Inputs

eod_controls
env_variables

1.33.2.2 Selection

env_variables
where code = ‘EOD_BY’

1.33.2.3 Processing

update eod_controls, setting TO_DATE to the system date
update env_variables, setting env_value to ‘EA3' where code = ‘EOD_BY’

run SQL script EA3_PROC_FI
run SQL script EA3_PRG_ARCHV
run SQL script <AGENCY_CODE>.SQL

1.33.2.4 Outputs

end_controls
env_varibles

Log file: EA3_SQL.LOG

1.34 Send redemption/rejection information to agencies

1.34.1 EA3_PROC_FI.SQL

Overview

Process the Food instrument's received from Central. Update the food instrument data of the local agency with the food instrument's received from the central server.

1.34.1.1 Inputs

a_i_food_instruments
a_i_fi_reject_reasons
i_food_instruments

1.34.1.2 Selection

For i_food_instruments:

WHERE serial_number = eodadm.a_i_food_instruments.serial_number

1.34.1.3 Processing

If a corresponding i_food_instruments record is not found for the a_i_food_instruments,
then write error message

Elseif i_food_instruments.date_modified = a_i_food_instruments.date_modified
and i_food_instruments.date_created = a_i_food_instruments.date_created (records are same)
then write error message.

Else Updates food instrument table from the food instrument information sent from the Central database:

```
UPDATE  i_food_instruments
SET     ou_seq_id=eodadm.a_i_food_instruments.ou_seq_id,
        revalidation_code=eodadm.a_i_food_instruments.revalidation_code,
        compliance_buy_flag=eodadm.a_i_food_instruments.compliance_buy_flag,
        ifit_fi_type_code=eodadm.a_i_food_instruments.ifit_fi_type_code,
        idis_disposition_code=eodadm.a_i_food_instruments.idis_disposition_code,
        vcc_case_id=eodadm.a_i_food_instruments.vcc_case_id,
        ifi_serial_number=eodadm.a_i_food_instruments.ifi_serial_number,
        bd_bank_disposition_code=eodadm.a_i_food_instruments.bd_bank_disposition_code,
        cfpp_effective_date=eodadm.a_i_food_instruments.cfpp_effective_date,
        cfpp_cc_client_id=eodadm.a_i_food_instruments.cfpp_cc_client_id,
```

ma_aty_act_type_code=eodadm.a_i_food_instruments.ma_aty_act_type_code,
ma_seq_nbr=eodadm.a_i_food_instruments.ma_seq_nbr,
ivr_void_reason_code=eodadm.a_i_food_instruments.ivr_void_reason_code,
cfpp_ifp_food_package_id=eodadm.a_i_food_instruments.cfpp_ifp_food_package_id,
ven_id=eodadm.a_i_food_instruments.ven_id,
ma_ven_id=eodadm.a_i_food_instruments.ma_ven_id,
cc_client_id=eodadm.a_i_food_instruments.cc_client_id,
sm_staff_member_id=eodadm.a_i_food_instruments.sm_staff_member_id,
fpo_po_number=eodadm.a_i_food_instruments.fpo_po_number,
stale_date=eodadm.a_i_food_instruments.stale_date,
issue_method=eodadm.a_i_food_instruments.issue_method,
est_reb_value=eodadm.a_i_food_instruments.est_reb_value,
first_date_to_use=eodadm.a_i_food_instruments.first_date_to_use,
last_date_to_use=eodadm.a_i_food_instruments.last_date_to_use,
maximum_amt=eodadm.a_i_food_instruments.maximum_amt,
redemption_amt=eodadm.a_i_food_instruments.redemption_amt,
cleared_date=eodadm.a_i_food_instruments.cleared_date,
reject_date=eodadm.a_i_food_instruments.reject_date,
requested_amt=eodadm.a_i_food_instruments.requested_amt,
approved_amt=eodadm.a_i_food_instruments.approved_amt,
approval_date=eodadm.a_i_food_instruments.approval_date,
issue_date=eodadm.a_i_food_instruments.issue_date,
void_date=eodadm.a_i_food_instruments.void_date,
obligated_amt=eodadm.a_i_food_instruments.obligated_amt,
formula_return_flag=eodadm.a_i_food_instruments.formula_return_flag,
allocated_amt=eodadm.a_i_food_instruments.allocated_amt,
missing_issuance_flag=eodadm.a_i_food_instruments.missing_issuance_flag,
process_date=eodadm.a_i_food_instruments.process_date,
date_modified=eodadm.a_i_food_instruments.date_modified,
modified_by=eodadm.a_i_food_instruments.modified_by,
note=eodadm.a_i_food_instruments.note
received_by_state=eodadm.a_i_food_instruments.received_by_state
cat_category_code=eodadm.a_i_food_instruments.cat_category_code
VCC_VEN_ID=eodadm.a_i_food_instruments.VCC_VEN_ID
CP2_ID=eodadm.a_i_food_instruments.CP2_ID
buy_number=eodadm.a_i_food_instruments.buy_number
issue_month=eodadm.a_i_food_instruments.issue_month
WHERE serial_number = eodadm.a_i_food_instruments.serial_number

If this FI is revalidated from Central then delete the corresponding reject reasons records from I_FI_REJECT_REASONS:

IF (eodadm.a_i_food_instruments.reject_date IS NULL) AND
(ifi_rec.reject_date IS NOT NULL) THEN
BEGIN
DELETE FROM i_fi_reject_reasons
WHERE ifi_serial_number = eodadm.a_i_food_instruments.serial_number

If this FI is rejected, then create respective records in the I_FI_REJECT_REASONS:

IF (eodadm.a_i_food_instruments.reject_date IS NOT NULL) THEN
BEGIN

```
                DELETE FROM i_fi_reject_reasons
                WHERE  ifi_serial_number = eodadm.a_i_food_instruments.serial_number

        INSERT INTO i_fi_reject_reasons(
                ifi_serial_number,
                irr_reject_reason_code,
                date_created,
                created_by,
                date_modified,
                modified_by)
                (SELECT
                        ifi_serial_number,
                        irr_reject_reason_code,
                        date_created,
                        created_by,
                        date_modified,
                        modified_by
                        FROM
                        eodadm.a_i_fi_reject_reasons
                        WHERE ifi_serial_number = eodadm.a_i_food_instruments.serial_number)
```

Delete the record from the Central-to-Agency transfer file, as it is no longer required:

```
DELETE FROM eodadm.a_i_food_instruments
WHERE serial_number=eodadm.a_i_food_instruments.serial_number;
```

Also delete reject reasons record from the Central-to-Agency transfer file:

```
DELETE FROM eodadm.a_i_fi_reject_reasons
WHERE  ifi_serial_number = eodadm.a_i_food_instruments.serial_number
```

1.34.1.4 Outputs

If the check is not found at the local agency then flash a message:

```
IF check record is not found THEN
        success := 'N';
        Output('Problem with check :  ' || eodadm.a_i_food_instruments.serial_number);
        Output('Problem finding the record here, Cannot be processed further');
        ELSE
```

If the check is found, but nothing has been modified (duplicate), then discard:

```
        IF TRUNC(eodadm.a_i_food_instruments.process_date) = TRUNC(ifi_rec.process_date)
        AND TRUNC(eodadm.a_i_food_instruments.date_modified) = TRUNC(ifi_rec.date_modified)
THEN success := 'N';

        OUTPUT('Duplicate Check : ' || eodadm.a_i_food_instruments.serial_number);

                ELSE

                EXCEPTION
                WHEN OTHERS THEN
                        OUTPUT('Serial Number : ' || eodadm.a_i_food_instruments.serial_number);
```

```
                    OUTPUT('Problem in Updating the Check');
                        success := 'N';

                    EXCEPTION
                    WHEN NO_DATA_FOUND THEN
                        NULL;
                    WHEN OTHERS THEN
                        success := 'N';
                        OUTPUT('Serial Number : ' ||
                        eodadm.a_i_food_instruments.serial_number);
                        OUTPUT('Problem in deleting the FI REJECT REASONS records');

                    EXCEPTION
                    WHEN NO_DATA_FOUND THEN
                        NULL;
                    WHEN OTHERS THEN
                        success := 'N';
                        OUTPUT('Serial Number : ' ||
                        eodadm.a_i_food_instruments.serial_number);
                        OUTPUT('Problem in inserting the FI REJECT REASONS records');
```

i_food_instruments
i_fi_reject_reasons

1.35 Transmit archival clients and update records at agencies

1.35.1 EA3_PRG_ARCHV.SQL

Overview

Purge archived client records from c_clients where a record exists in r_archived_clients for that client ID. The deletes have been ordered from the smallest level child up to the parent (c_clients). Archived vendors do not get purged here as that process is handled through delete triggers.

1.35.1.1 Inputs

r_archived_clients

1.35.1.2 Selection

All records in the r_archived_clients table.

1.35.1.3 Processing

Purges archived client records from the following tables where the client_ID matches the client_ID in r_archived_clients.

LEVEL 5:
A_APPT_TOPIC_MATERIALS

LEVEL 4:
A_APPT_TOPICS
A_APPT_APPT_ITEMS
AAS_APPT_CLIENTS
C_CLIENT_SVC_NE_MATERIALS
F_WAIT_LIST_CONTACTS
C_DIET_NUTRIENTS
C_INCOMES

LEVEL 3:

A_APPOINTMENTS
C_I_C_HH_REASONS_BFENDS
C_W_HH_REASONS_BFENDS
C_FOOD_BOX_DISTS
C_INCOME_HISTORIES
C_CLIENT_NE_TOPICS
C_WOMAN_MEDICALS
C_BLOODWORK_DATA
C_CERT_TERM_REASONS
C_B_N_HEALTHS
C_FOOD_PACKAGE_PRESCRIPTIONS
F_WAIT_LISTS
C_P_HEALTHS
C_INFANT_CHILD_MEDICALS
C_DIETARY_ASSESSMENTS
C_HEALTH_RISK_FACTORS
C_I_C_HEALTHS
C_ECONOMIC_UNIT_MEMBERS
C_PEER_RBFENDS
C_CONTACTS
C_P_COUNS_NOTES
I_BF_VCHR_ITEMS

LEVEL 2:
I_BF_PROMO_VCHRS
V_COMPLIANCE_CASE_CLIENTS
C_CLIENT_SERVICES
C_CERTIFICATIONS
C_TRANSFER_HISTORIES
C_IMMUNIZATIONS
C_CLIENT_REFERRALS
C_CLIENT_COMMUNICATIONS
C_RESOLUTIONS
C_CLIENT_NOTES
C_CLIENT_PROGS
C_CLIENT_GROUPS
C_INCOME_HISTORIES
C_INFANT_DATA
C_CLIENT_GOALS
C_CERT_PEER_COUNSELS
C_WOMAN_MEDICALS
C_W_HEALTH
C_INFANT_CHILD_MEDICALS
C_I_C_HEALTHS
C_BLOODWORK_DATA
C_PREV_FAMILIES
C_PREV_NAMES
C_ALLERGY_FOODS
C_MORE_ETHNIC_GROUPS

LEVEL 1:

AAS_APPT_CLIENTS
AAS_APPT_ITEMS
AAS_APPOINTMENTS
AAS_CLASS_FAMILIES
C_CLIENTS
C_INCOMES
C_ECONOMIC_UNIT_MEMBERS
C_INCOMES_HISTORIES
DEL_STATEMENTS

1.35.1.4 Outputs

N/A

1.35.2 EA3_RETRIEVE.SQL

Overview

Run scripts EA3_INS_RETR_CLI.SQL and EA3_INS_RETR_VEN.SQL to retrieve clients and vendors who were archived and then requested to be put back on the system.

1.35.2.1 Inputs

N/A

1.35.2.2 Selection

N/A

1.35.2.3 Processing

N/A

Run scripts EA3_INS_RETR_CLI.SQL and EA3_INS_RETR_VEN.SQL

1.35.2.4 Outputs

1.35.3 EA3_INS_RETR_CLI.SQL

Overview

This script performs inserts from the temporary tables into all client tables.

r_c_clients
r_c_bloodwork_data
r_c_w_healths
r_c_w_hh_reasons_bfends
r_c_certifications
r_c_cert_peer_counsels
r_c_cert_term_reasons
r_c_client_communications
r_c_client_goals
r_c_client_ne_topics
r_c_client_notes
r_c_client_progs
r_c_client_referrals
r_c_client_services
r_c_client_svc_ne_materials
r_c_contacts
r_c_dietary_assessments
r_c_diet_nutrients
r_c_economic_unit_members
r_c_family_economic_units
r_c_family_phones
r_c_fam_referrals
r_c_feu_communications
r_c_health_risk_factors
r_c_incomes
r_c_income_histories
r_c_infant_child_medicals
r_c_infant_data
r_c_i_c_healths
r_c_i_c_hh_reasons_bfends
r_c_peer_rbfends
r_c_p_couns_notes
r_c_p_healths
r_c_resolutions
r_c_transfer_histories
r_c_woman_medicals
r_s_client_archives
r_c_smoking_histories
r_c_prev_families
r_c_more_ethnic_groups
r_c_prev_names
r_c_BF_promo_issuances

r_c_allergy_foods
r_c_PS_responses
r_c_client_BP_issuances
r_c_resolved_clients
r_c_cert_nutr_eds
r_c_food_box_dists
r_c_food_package_prescriptions

1.35.3.1 Inputs

r_s_client_archives

1.35.3.2 Selection

where local_agency in
(select env ('AGENCY_SEQ') from dual)

1.35.3.3 Processing

This calls database function ins_arch_cli which recreates all of the client's information. It returns a "0" if successful, a "1" if the client already exists, a "2" if no information exists, and a "3" for other errors.

1.35.3.4 Outputs

```
if ret_val = 0 then
        commit;
        OUTPUT('Successfully Retrieved Client '||to_char(cl_id));
else rollback;

if ret_val = 1 then
        OUTPUT('Client '||to_char(cl_id)||' already exists, Client not Processed' );
        elsif ret_val = 2 then
                OUTPUT('Client '||to_char(cl_id)||' does not exist in Temporary Tables, Client not
                Processed' );

                elsif ret_val = 3 then
                        OUTPUT('Client '||to_char(cl_id)||' generated Oracle Error, Client not Processed'
                        );
end if
```

1.35.4 EA3_INS_RETR_VEN.SQL

Overview

This script performs inserts from the temporary tables into all vendor tables.

1.35.4.1 Inputs

r_s_vendor_archives

1.35.4.2 Selection

N/A

1.35.4.3 Processing

This calls database function ins_arch_ven which recreates all of the vendor's information. It returns a "0" if successful, a "1" if the vendor already exists, a "2" if no information exists, and a "3" for other errors.

1.35.4.4 Outputs

```
if ret_val = 0 then
        commit;
```

```
        OUTPUT('Successfully Retrieved Vendor '||to_char(ven_id));
else
        rollback;

if ret_val = 1 then
        OUTPUT('Vendor '||to_char(ven_id)||' already exists, Vendor not Processed' );

        elsif ret_val = 2 then
                OUTPUT('Vendor '||to_char(ven_id)||' does not exist in Temporary Tables, Vendor not
                Processed' );

                elsif ret_val = 3 then
                        OUTPUT('Vendor '||to_char(ven_id)||' generated Oracle Error, Vendor not
                        Processed' );
end if;
```

1.36 Print out status logs of the process

1.36.1 EA_PRINT.BAT

Overview

This batch file produces log files by agency for the following processes:

* Insert (new / updated data going to central)
* Export
* Truncate
* Import
* Delete
* Insert (new data from central)
* Update

1.36.1.1 Inputs

N/A

1.36.1.2 Selection

Called by EA3.BAT

1.36.1.3 Processing

The system sets the environmental variables for the particular Local Agency reading them from the EA1.INI file and places them in a temporary file to be used for retrieving scripts.

Creates combined log files for the following files:

1. Combines the EA.LOG with the EA_HIS.LOG by writing it to the end of the EA_HIS.LOG file.

2. Writes the EA1_SQL.LOG, EA3_SQL.LOG, EA3_RETRIEVE.LOG, and EA4_SQL.LOG files to the end of the AGCY_SQL.LOG.

3. Then creates files AGCY_CTRL.LOG (new file written each day) AGCY_%.LOG (contains the history of agcy_ctrl.log updated each day) that log the insert and export information that is dumped from the Agency to Central. The following logs are written to the end of the AGCY_CTRL.LOG and AGCY_%.LOG (the wildcard represents the Local Agencies ID):

The insert section: EA1_TAB_UP.LOG
The export section: EXP.LOG
The EOD export section: EODEXP.LOG

4. Creates the files CTRL_AGCY.LOG (new file written each day) and AGCY_%.LOG (contains the history of agcy_ctrl.log updated each day) that log the truncate, import, delete, insert, and update information that is loaded from Central to the Local Agency. The following logs are written to the end of the CTRL_AGCY.LOG and AGCY_%.LOG (the wildcard represents the Local Agencies ID):

The truncate section: EA3_TRNC.LOG
The import section: IMP.LOG
The delete section: EA3_DELTRIG.LOG
The insert section: EA3_IN_TAB.LOG
The update section: EA3_TAB_UP.LOG

The final combined files are EA_HIS.LOG, AGCY_CTRL.LOG, CTRL_AGCY.LOG, and AGCY_%.LOG. Examples of these log files can be found in Appendix A.

1.36.1.4 Outputs

Combined output of agcy_ctrl.log, agcy_%.log, and ctrl_agcy.log.

1.37 RUN_CASELD.CMD

Overview

This command file begins monthly caseload processing at the State level.

1.37.1 Processing

This script runs at the start of every month on the central FTP Server. It calls another script named: INSERT_MONTHLY_CASELOADS which populates caseload information for the current month on the State database.

1.37.2 INSERT_MONTHLY_CASELOADS.SQL

Overview

This SQL script populates the caseload information for the current month on the State database and updates the caseload information for the past 2 months for the voided checks. It executes caseload population package's NEW_FCTD procedure to insert new caseloads for the current month and insert_only_participant procedure to update the past one-month's caseloads. It also runs a procedure to purge old caseload type details records, which are 42 months old and put it in the history table.

It also runs a package to create a monthly csfp caseloads process called caseload_csfp.new_fctd.

At the start of the month this script will first populate the F_CASELOAD_TYPE_DETAILS table with the Client's ID, Category Code, Language Code, Ethnic Group Code, Priority ID, Clinic Sequence ID, the current Fiscal Month and Year, the Client's Income, Family Size, Adjunctively Eligible Flag, Migrant Flag, Refugee Flag, State Funding Flag, and WIC/CSF Program. The logic to populate caseloads tables is as follows:

1. Enrollment Logic: The certified_flag will be set to Yes if

- The client will be considered as enrollee if:
  - The client is certified for least one day in the month:
    So If the client's cert_start_date <= last day of month and
    Client's cert_end_date >= first day of month and
    (1.4) first day of month – cert_start_date <= 60 and
    Cert_Termination_date is null or >= first day of month
    Client is certified under the WIC program.
- A no-show is a client who is certified and has not participated (this is not related to missing appointments). As per USDA, a client who is no-show for 2 or more months may be removed from the enrollment count. After two months of no-shows the client should not be counted in the enrollee total. We will not change the certification table but make his change on the report side. This creates a distinction between "Certification" and "Enrollment". Certification is what is in the Certification table, and Enrollment is a subset of Certification, which excludes no-shows. Also, Certification occurs by the day and Enrollment occurs by the month.

- No-Show = certified but not participating for 60 days 4(Not really participation because participation is by the month. The client is certified by does not receive a valid FI for 60 days.) Out of state transfers can't be checked for no show clients since AIM will not have the history of food instruments received from other states.
- If a client is transferred within a certificated month, then the client is counted as enrolled in the last clinic they are certified in for that month.
- For the category changed clients consider original cert start date.
- If the client doesn't have a valid income, then check for the dual enrollment in c_resolutions table , if a record is found in the dual enrollment table then consider it as a dual enrollment and don't report the client for caseloads.

2. Participation Logic: The participant flag will be set to Yes if

The client should be considered as a participant if:

- The client has received Food instruments for the reporting month and at least one food instrument is not voided.

OR

- If all the food instruments for the client in the reporting month are voided, but

a. There exists at least one FI with one of the following void reasons (No matter if the void date is before, equal to, or after first date to use)

B - Lost Or Stolen
E - Returned By Client/Unused
Z - Stale Dated By EOD
L - Signed Before Going To Store

OR

b. There exists at least one food instrument voided after the first date to use with one of the following void reasons

D - Replaced/Food Package Change
H - Termination
J - Late Pickup/Reduced Food Package

The client should be considered as a Non-participant if:

A.

If the client has all the Food Instruments voided for the reporting month with one of the following reasons; and there is no second reissued set of food instruments.

C - No-Show/Unclaimed
A - Damaged, Unissued

F - Printed/Not Issued
G - Misprinted Text
I - Categorical Ineligibility

B.

- **I**f a participant is breastfeeding an infant (Infant category IEN or IPN), then that infant is also a WIC participant.
- If the breastfeeding infant's mother is a participant as per above logic then count the IPN/IEN as a participant too.
- If there is no linked mother for the IEN or if the mother is not a participant, then check if IEN has been assigned a dummy food package 'AA900888' else use the above logic for the infant to determine their participation.

C.

If a client transfers, then the client is counted as a participant in the last clinic they participated in for the month in question.

Income-related Case Exceptions for Participation and Enrollment

- All exceptions should be reported as an "Other" category. Any data problems in report fields such as the client's race/ethnicity, poverty level, family size, and income eligibility should be reported as "Other". Since the family_size and income defines the poverty level, creation of a new "Other" category for just the poverty level should handle the exceptional cases for income or family size too. We will create a new "Other" category for race ethnicity too in order to handle undefined race ethnicity cases. None of these report fields can exclude a client from participation; however participants should not have exceptions in required fields. If exceptions occur they need to be documented and investigated.
- If the client's income changes to income ineligible, then the client should return all the remaining FIs and should be terminated from certification. As a practical matter, the client may retain valid FIs for the current month and up to the next two following months.
- Currently AIM doesn't have automatic termination facility because the client can have valid food instruments issued, so those FI's need to be returned by client and need to manually void from AIM and terminate the client. But if the client doesn't return unused FI's then the client will still be a participant.
- If a client's certification is terminated but they still have valid FIs for future months, then the client will be a participant but not an enrollee. This should be very rare.
- For the Out of State transfers, since the income entry is not mandatory, it should be reported under "Other Income" category. The report should have comments indicating the possible reasons for any client in an "Other" category.

If a valid income record is not found between the certification start date and certification end date for the client then the income "Other" category will be used to report participation.

On every EOD run,
new F_CASELOAD_TYPE_DETAILS records are populated for all Clients who have already been issued checks for the new month, as well as any Exclusively Breastfeeding Infants who still have a mother on WIC. When Food Instruments are created for existing Clients, the F_CASELOAD_TYPE_DETAILS.PARTICIPANT_FLAG is updated for all applicable records.

It will also generate the F_CASELOADS.CLIENT_COUNT by incrementing this value by 1 for each record in the F_CASELOAD_TYPE_DETAILS table for the Category Code, Language Code, Ethnic Group Code, Priority ID, State Funding Flag, Clinic Sequence ID, the current Fiscal Month and Year, and WIC/CSF Program where F_CASELOAD_TYPES.DESCRIPTION = 'ENROLLEE'. The count is repeated for clients where the F_CASELOAD_TYPE_DETAILS.MIGRANT_FLAG = 'Y', updating the F_CASELOADS table where F_CASELOAD_TYPES.DESCRIPTION = 'MIGRANT'. The same process is then followed for Clients where the F_CASELOAD_TYPE_DETAILS.PARTICIPANT_FLAG = 'Y', updating the F_CASELOADS table where F_CASELOAD_TYPES.DESCRIPTION = 'PARTICIPANT'.

3. CSFP Enrollment Logic :

Pick all the csfp clients with a valid certification for that month.
A valid certification is with cert_start_date <= Last day of that month,
And cert_end_date >= 1st day of that month
And (termination_date is null
Or
(termination_date >= 1st day of that month)

Pick up the most recent income information entered for these clients from c_income_histories table for either the family_id or client_id where the income_date between cert_start_date and cert_end_date. If there is no income record in c_income_histores table then default the income to the income from c_family_economic_units table.

Based on this income and family_size get the poverty level id. For the adjunct income eligible clients get the poverty level id with low_income = 0 and high_income = 0 too.

For the out of state transfer clients the clients will get enrolled with the poverty id as for the adjunct eligibility currently, since their income is 0.

If for some reason, there is no associated income with that client then that client won't be considered as a caseload candidate, since income is necessary to get proper poverty id.

If the client is transferred in a month then he will be counted as enrolled in the most recent agency/clinic he is being transferred.

Participant:

If the client has picked up at least one valid food box in the month then he is considered as a participant. If the client is transferred in a month then he will be counted as participated in the agency/clinic where he has received the food box.

1.37.3 Outputs

F_CASELOADS_CSFP

F_CASELOAD_TYPE_DETAILS_CSFP

1.38 Monthly population of participation information for farmer market

1.38.1 Overview

The AIM System compiles the caseload participation information for the clients that were redeemed the coupons in the past and for those whose coupons were created in the past months. During the compilation process, the system processes the information and inserts it in caseload tables. The following tables are updated:

M_caseload_type_details
f_caseloads

1.38.2 Participation Information

(Tables are f_caseloads and m_caseload_type_details)

EOD Script is "ec2_populate_clinic.sql"

The key fields in m_caseload_type_details table are as follows :

Cc_Client_Id - Client_id from c_clients
Ou_Seq_Id – the agency/clinic they are enrolled in from c_family_economic_units table
Fbu_Ffy_Month – Fiscal month
Fbu_Fff_Ffy – Fiscal Year
Cat_Category_Code – Category code from c_certifications table
Ceg_Ethnic_Group_Code – Ethnic Group code from c_clients
Lang_Language_Code – Primary language code from c_family_economic_units table
Cp2_Id – Priority from c_certifications table
Program – The Program under which they are certified i.e. 'WIC'
State_Fund_Flag – State fund flag from certifications table
Date_Created – Date on which caseload record is created
Created_By – Database User who created this record
Fpl_Id – Determine fpl_id based on the family's income from f_poverties_to_incomes table
Certified_Flag – This flag will be 'Y' i.e. Yes if the client is enrolled
Participant_Flag - This flag will be 'Y' i.e. Yes if the client is a participant
Migrant_Flag - This flag will be 'Y' i.e. Yes if the client is Migrant
Adj_Eligible_Flag – Client's adjunct eligibility flag
Refugee_Flag - This flag will be 'Y' i.e. Yes if the client is Refugee
Income – Family's Recent Income in that fiscal month
Family_Size – Total number of family members from c_family_economic_units table
Date_Modified - Date on which caseload record is last modified
Modified_By - Database User who modified this record last

The Rollup table from type details table is f_caseloads table , it has the sum of all the caseload client in the client_count field grouped by category_code, ethnic_group_code, agency/clinic, language, povery_id, fiscal_month, fiscal_year, priority, program. The structure of this table is as follows :

Caseload_Id – Unique identifier
Cat_Category_Code – Category code

Ceg_Ethnic_Group_Code – Ethnic Group Code
Ou_Seq_Id – Agency/Clinic
Lang_Language_Code – Language Code
Fpl_Id – Poverty Id
Fbu_Ffy_Month – Fiscal Month
Fbu_Fff_Ffy – Fiscal Year
Fc_Caseload_Type_Code – The type code will determine if the record gives total enrollee, participant, migrant or refugee count
(8- farmer market Participant, 7- Farmers Market Enrollee)
Cp2_Id - Priority
Client_Count – Total count of clients
Program - WIC
State_Fund_Flag – State fund flag from certifications table
Date_Created – Date on which caseload record is created
Created_By – Database User who created this record
Bf_Inf_Count – Never used
Date_Modified - Date on which caseload record is last modified
Modified_By - Database User who modified this record last

The logic to populate caseloads table:

A new procedure will be added in the caseload package to take care of participation for farmers' market client which will use m_caseload_type_details and f_caseloads tables.

Caseload. Insert_caseload_type_farmers:
This caseload procedure will run during EOD process and will mark the clients who received the coupons for that day as enrolled clients for farmers market. Type code is "7".

Caseload. Update_caseload_type_farmers:
This caseload procedure will run when we receive a paid file for farmers market from FSMC, and it will mark the enrolled clients as the participant for the month in which their first coupon is redeemed. Type code is "8".

E.g.: If client C1 has been issued the coupons in January, but found a redeemed coupon for the first time in March then client will be considered as a participant for March. If another coupon is redeemed in April, the client is not counted again as a participant.